Visualización de valencia (Pleasure-Dominance-Arousal) en el dispositivo MUSE

Javier Bonet Lambea

Directora: Rosa Mª Gil Iranzo

29 de junio de 2017

AGRADECIMIENTOS

En primer lugar quiero dar las gracias a mi tutora, Rosa Gil, que fue quien propuso la idea para este proyecto, y me ha ayudado durante todo el proceso.

Al profesor Juan Miguel Gil de la UPV por la aportación de imágenes de la IAPS con las que realizar el experimento.

A mis compañeros, que siempre han estado ahí cuando he necesitado algo. En especial

A mi familia, por el apoyo que me han brindado durante todo este tiempo.

RESUMEN

En los últimos años, el progreso tecnológico de los dispositivos móviles, ha propiciado su popularización, hasta el punto en que resulta complicado imaginar imaginar un día a día sin smartphones. Debido al gran crecimiento y desarrollo de estos dispositivos, encontramos que cada vez disponen de mayores prestaciones, lo que aumenta su utilidad.

Por otra parte, en el ámbito biotecnológico la popularización de los smartphones ha generado un nuevo mercado: dispositivos biométricos compatibles con los mismos. De esta forma, disponemos de aparatos portátiles con características que hasta ahora solo se encontraban en instrumentos de laboratorio. Estos nuevos dispositivos están enfocados al uso del gran público y adaptados para ser utilizados en cualquier lugar.

Este proyecto tiene como objetivo desarrollar una herramienta sencilla de utilizar para el usuario y con la que poder visualizar fácilmente cómo responde nuestro cerebro ante distintos estímulos a través de datos obtenidos mediante un sensor EEG de bajo coste.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

Las aplicaciones móviles cada vez tienen más importancia en nuestras vidas. Por otra parte, cada vez están surgiendo más dispositivos biométricos. Con ellos es posible monitorizar varios aspectos de nuestro cuerpo. Un tipo de estos dispositivos biométricos, conocidos como BCI (Brain Computer Interface), es en el que nos vamos a centrar durante el desarrollo de esta aplicación.

Estos dispositivos se valen de la actividad eléctrica cerebral para registrar datos mediante la electroencefalografía. En esta aplicación, vamos a recopilar estos datos y a ofrecer una forma sencilla de visualizarlos.

1.1 OBJETIVOS

El objetivo principal del proyecto es desarrollar una aplicación para dispositivos móviles Android, con la que sea posible visualizar y guardar información proveniente de dispositivos que captan señales biométricas (ondas cerebrales, en nuestro caso) en tiempo real y en cualquier situación, no solo en un laboratorio. Y además, poder observar cómo varían las ondas cerebrales cuando somos sometidos a distintos estímulos.

Además de esto, otros objetivos del trabajo son:

- Comprender las ondas cerebrales.
- Comprender y aplicar el procesado de señales digitales.
- Manejar la conexión entre un dispositivo Android y un dispositivo externo vía Bluetooth.
- Crear una aplicación robusta y usable.
- Comprobar la fiabilidad de las mediciones en exteriores.

1.2 MOTIVACIÓN

Al empezar a pensar en un tema para el TFG, y tras haber participado en el desarrollo de algunas *apps* móviles, tenía claro que quería desarrollar una app, pero no sabía hacia donde enfocarla.

Este proyecto surge de la propuesta de mi tutora, y me motivó principalmente por dos razones: requería adentrarme en un campo que era totalmente desconocido para mi, las ondas cerebrales; y me permitiría aprender como funcionan tecnologías que hasta ahora no había utilizado.

Finalmente, la motivación más importante viene del hecho de ser capaz de gestionar un proyecto desde su inicio, desarrollarlo, y poder obtener unos resultados a partir de una herramienta creada por mi mismo.

1.3 TEMPORALIDAD / PRESUPUESTO

1.3.1 Temporalidad

La gestión del tiempo en un proyecto es una de las partes más importantes del mismo, que suele quedar relegada a un segundo plano. Como el presupuesto se contabiliza en horas de trabajo, para poder realizar una estimación, dividiremos el desarrollo en tareas.

El proyecto comienza en febrero y establecemos su fecha de entrega en junio, por lo que el tiempo disponible para su realización son 20 semanas.

Una vez sabemos el tiempo disponible y las tareas a realizar, procedemos a repartir el tiempo entre las mismas, para garantizar que el proyecto va a poder ser llevado a cabo.

Tarea	**Planificación**	**Coste real**
Fase de inicio	6	4
Lecturas previas (EEG, DSP)	4	2
Identificar necesidades de la aplicación	1	1
Familiarización con SDK Muse	1	1
Fase de Desarrollo	8	10,5
Diseño	2	2,5
Implementación	6	8
Fase de Finalización	2	2
Pruebas con usuarios	2	?
Documentación	-	4

Tabla 1.: Planificación de tareas

Tras la elaboración del proyecto, podemos observar que la duración final se ha ajustado a la estimada, aunque algunas de las tareas se han alargado durante más tiempo del previsto. La duración de otras tareas, en cambio, ha sido menor de lo estimado. Además, en un principio no había tenido en cuenta en la planificación del tiempo una parte del proyecto muy importante y que ha sido una

de las tareas que más tiempo ha requerido, la documentación del mismo.

1.3.1.1 *Diagrama de GANT*

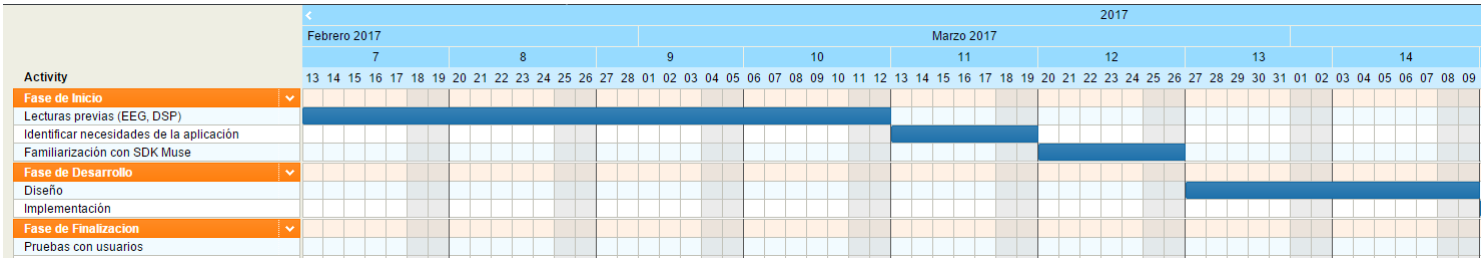


Figura 1.: Diagrama de gant. Parte 1.

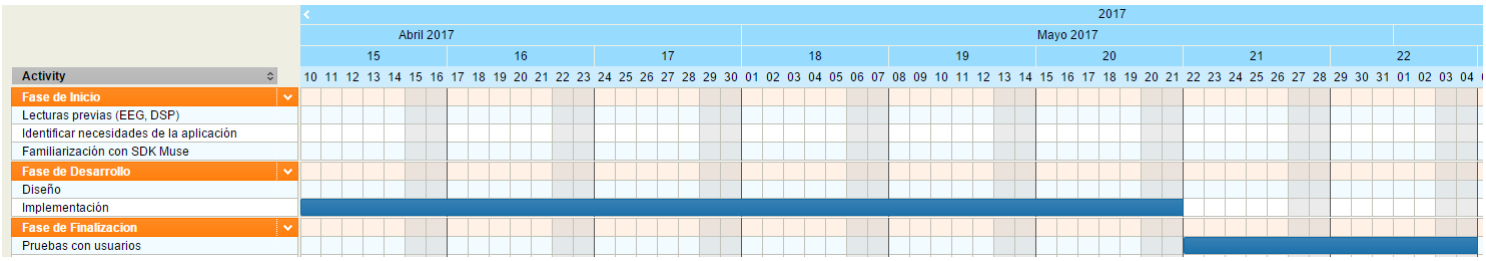


Figura 2.: Diagrama de gant. Parte 2.

1.3.2 Presupuesto

Respecto al presupuesto, requiere un desembolso en el dispositivo EEG utilizado para las pruebas con usuarios. Si fuera necesario un dispositivo Android por no disponer de alguno, también debería contabilizarse como un gasto del proyecto.

Por otra parte, habría que contabilizar los costes de mantener a un trabajador, dedicando una media de 20h a la semana durante las 20 semanas establecidas anteriormente en la planificación. Finalmente, utilizamos la versión gratuita de Firebase, pero en caso de necesitar más capacidad, habría que añadir sus costes al presupuesto.

Así, el presupuesto quedaría de la siguiente forma:

Tarea	**Precio**	**Cantidad**
MUSE Headband	299€	1
Dispositivo Android	150€	1
Trabajador	12€/h	400
Total	5249€	

Tabla 2.: Presupuesto

Tras elaborar el presupuesto, obtenemos que el coste de realizar el proyecto sería de 5000€ apróximadamente, lo cual es un coste que se acerca bastante al coste de desarrollo de un proyecto comercial de estas carcterísticas.

2 ELECTROENCEFALOGRAFÍA (EEG)

Este capítulo explica qué es y cómo funciona la tecnología EEG, que es la base sobre la que se apoya la aplicación, y las diferentes ondas generadas por la actividad cerebral.

2.1 REGISTRANDO LA ACTIVIDAD CEREBRAL

El cerebro es un órgano muy complejo, formado por millones de neuronas tejiendo una compleja red de comunicación que utiliza la electricidad como forma de transmitir la información. Cada comunicación genera un pulso eléctrico y de la combinación de estos pulsos, surgen las ondas cerebrales. Conocemos como electroencefalografía (EEG) al registro de estas ondas. Fue aplicado en humanos por primera vez en 1924 por Hans Berger.

Los datos que se registran provienen de la actividad generada al comunicarse dos neuronas entre sí. Dicho registro varía mucho con la localización de los electrodos. Esto es debido al gran número de conexiones que presentan las neuronas. Por ello, dependiendo de cual sea la naturaleza del estudio, y de los datos que se pretenden obtener, se colocan unos electrodos u otros.

Para determinar la ubicación de los electrodos existen diversos métodos, como el sistema 10-20 representado en la **Figura** 3. De la necesidad de utilizar más electrodos, surgen otros sistemas derivados del sistema 10-20, añadiendo los electrodos entre los que ya define este sistema.

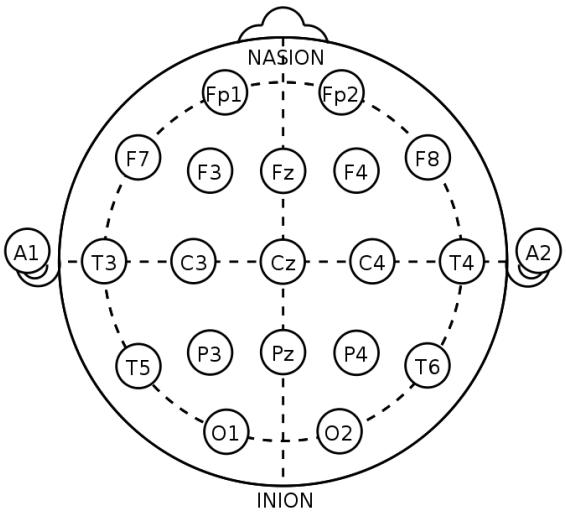


Figura 3.: El sistema 10-20, método estándar de posicionamiento de los electrodos en el EEG

2.2 IDENTIFICANDO LA ACTIVIDAD CEREBRAL

En 1930, Hans Berger realizó numerosos estudios, con los cuales preparó un informe. En este informe se designó por primera vez con caracteres del alfabeto griego a los tipos de onda observados durante los registros. Estas ondas observadas fueron: *Alfa*, las de menor frecuencia y *Beta* las de mayor.

Desde que fueran realizados los primeros estudios hasta hoy, se ha avanzado notablemente en el campo de la electroencefalografía y ahora podemos hablar de varios tipos de ondas más de las que inicialmente Berger propuso.

2.2.1 Las ondas cerebrales

Analizando el espectro de frecuencias y el comportamiento de las ondas cerebrales, se han identificado de la siguiente forma:

- Ondas **delta**: Es el rango de frecuencias de 0 a 4 Hz. Tiende a tener la amplitud más alta y a ser la onda más lenta. Se manifiesta normalmente durante el sueño. En otras circunstancias, puede ser indicador de lesiones cerebrales.

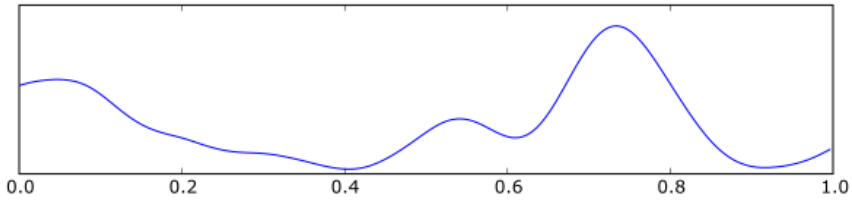


Figura 4.: Ondas delta

- Ondas **theta**: Las ondas theta se encuentran en el rango de frecuencias de 4 a 8 Hz. Este tipo de ondas se manifiesta de forma más común en niños, pero también puede encontrarse en adultos en estado de meditación o como respuesta a ciertos estímulos. Se asocia a estados de relajación y creatividad.

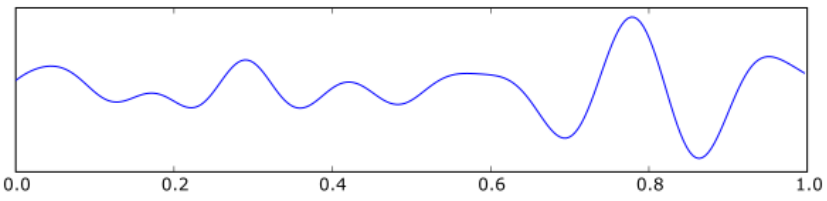


Figura 5.: Ondas theta

- Ondas **alfa**: Es el rango de frecuencias de 8 a 14 Hz. Fueron descubiertas por Berger que calificó como la onda alfa al primer tipo de onda EEG que consiguió registrar. Estas ondas se generan mayormente mientras se está despierto, con los ojos cerrados. Se reducen considerablemente con los ojos abiertos y durante el sueño.

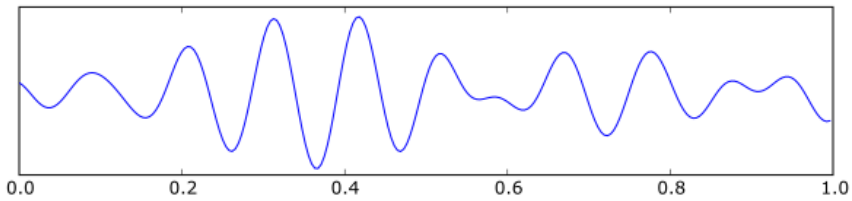


Figura 6.: Ondas alfa

- Ondas **beta**: En el rango de 14 a 30 Hz encontramos las ondas beta, descubiertas por Berger al mismo tiempo que las alfa. Están asociadas a un estado de actividad, pensamiento o concentración. Debido a su amplio rango, suelen dividirse en sub-ondas para su estudio.

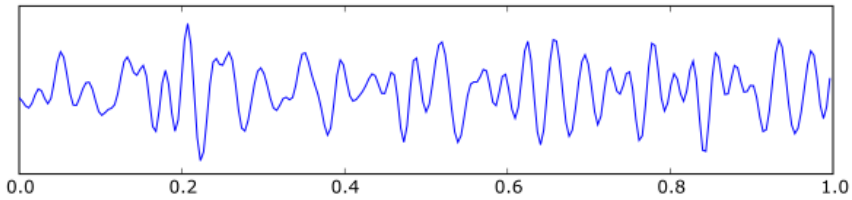


Figura 7.: Ondas beta

- Ondas **gamma**: Se califica como ondas gamma a la actividad encontrada a partir de los 30 Hz. Es la onda de la que menos datos se conocen, pues fue descubierta de forma mucho más tardía que el resto. Esto se debe a que la electroencefalografía analógica no era capaz de detectar las frecuencias de la onda gamma, y hasta que no surgió la digital, no pudo ser observada.

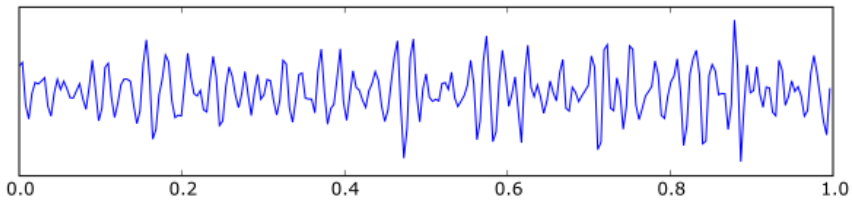


Figura 8.: Ondas gamma

2.3 APLICACIONES DE LA EEG

La electroencefalografía tiene un gran número de usos y aplicaciones bastante diversas entre sí, y cada vez se utiliza en más campos. Algunas de estas aplicaciones son:

- Monitorizar estado de alerta, coma y muerte cerebral.
- Localizar áreas dañadas tras una lesión en la cabeza, como derrames o tumores.
- Detectar emociones o estímulos.
- Monitorizar el aprendizaje.

- Detección de enfermedades mentales.

2.4 DETECCIÓN DE ESTÍMULOS

Una de las aplicaciones que tiene la tecnología EEG es la detección de estímulos. Actualmente se utilizan diversos métodos para detectar estos estímulos, en función del resultado que se quiera obtener.

2.4.1 Onda P300

Uno de los métodos que se utilizan es la Onda P300. Se conoce como Onda P300 al potencial evocado que aparece con una latencia de entre 250 y 500ms tras un estímulo. De ahí obtiene su nombre, ya que suele aparecer a los 300ms.

Su aparición se debe a la reacción cuando el sujeto está enfocado en responder a un estímulo en concreto (target). En cambio, aunque el sujeto reciba una serie de estímulos, si no son los que está esperando recibir (non-target), la onda P300 no aparece. En la Figura 9 podemos observar el aspecto de dos mediciones, una con el sujeto esperando un estímulo y reaccionando al mismo; y en la otra recibiendo una serie de estímulos que son ignorados.

Sus aplicaciones varían desde ser utilizada en detectores de mentiras hasta detección de enfermedades mentales.

Figura 9.: Onda P300: Comparación respuesta a estímulo target y non-target

2.4.2 Asimetría EEG frontal

La asimetría frontal consiste en comparar la activación entre dos lugares homólogos del cerebro. Como por ejemplo, comparar la activación entre los electrodos AF7 y su homólogo AF8, correspondientes al hemisferio izquierdo y derecho, según la especificación del sistema 10-20, vista en la Figura 3 en la página~5. Diversos estudios han utilizado la asimetría frontal, sugiriendo la posibilidad de que esta sea un indicador de la activación cerebral y de la respuesta a estímulos.

Estos estudios se pueden clasificar en varios campos, como los que se centran en el humor y en cómo afectan los estímulos al mismo, o los que estudian cómo afecta la asimetría frontal a la memoria emocional. En uno de sus estudios, Davidson[1] propone que es el hemisferio izquierdo el que procesa los estímulos positivos, mientras que los estímulos negativos son procesados por el derecho.

2.4.2.1 *Asimetría frontal alfa*

Dentro de la asimetría frontal, podemos diferenciar varias asimetrías, dependiendo del tipo de onda que se utilice para medirla. En este caso, como su nombre indica, la asimetría frontal alfa proviene de capturar la asimetría en las ondas alfa.

Está bastante aceptado que la relación entre la emisión de ondas alfa y la activación de la zona cerebral es inversa. Es decir, conforme la emisión de ondas alfa en una región en particular disminuye, la actividad cerebral en la misma aumenta.

Para comparar la actividad en ambas regiones, no nos fijamos en la intensidad absoluta de la señal, si no en la intensidad relativa entre las dos señales obtenidas. Uno de los cálculos que se utiliza para establecer la relación entre la señal de ambas regiones es:

$$\text{Índice de Asimetría frontal} = \log\left(\frac{\text{valor derecho}}{\text{valor izquierdo}}\right)$$

De esta forma, cuando las dos señales son iguales el indice de asimetría frontal es 1. Aunque esto es un caso muy poco frecuente, ya que por norma general un hemisferio está más activo que el otro. En este caso, si valor derecho $>$ valor izquierdo, el índice de asimetría será > 0, en caso contrario será < 0.

Debido a la naturaleza de este valor, para su obtención es necesario que el dispositivo utilizado cuente con un mínimo de dos electrodos ubicados simétricamente en la parte frontal, como pueden ser Fp1 y Fp2 o AF7 y AF8.

Este es el tipo de detección de estímulos que vamos a utilizar para determinar la actividad del usuario que utilice la aplicación.

3 ESTADO DEL ARTE

En este capítulo se va a analizar el mercado para conocer las diferentes tecnologías disponibles, así como la escogida para el desarrollo de la aplicación. También se van a comentar estudios relacionados con la materia.

3.1 ESTUDIOS REALIZADOS

Para poder realizar este trabajo he tenido que comprender qué son las ondas cerebrales, cómo se captan e interpretan, y cómo se trata esta información una vez digitalizada.

Para poder captar las ondas generadas por el cerebro, se utiliza la electroencefalografía, que ya ha sido explicada en el capítulo 2. Es el método a través del cual se obtienen las mediciones utilizadas por la aplicación.

Las ondas cerebrales como su propio nombre indica, no son otra cosa que ondas, por ello para ser capaz de comprenderlas, primero he tenido que aprender cómo funciona el Procesamiento de Señales Digitales (DSP, por sus siglas en inglés).

El *DSP* es un conjunto de procesos a través de los cuales transformamos la señal original para conocer más datos sobre ella. Uno de los más importantes es la Transformada de Fourier, que convierte la señal del dominio del tiempo al dominio de la frecuencia. Esta transformada va a ser la que utilizaremos durante el procesamiento de los datos, para generar el espectrograma, entre otras cosas.

En procesamiento de señales, para evitar las discontinuidades entre una muestra de FFT y la siguiente, se utilizan funciones conocidas como Ventanas. Para el EEG, la más utilizada suele ser la Hamming Window. Esta función de ventana es comentada más adelante, en la sección 4.2.5.1 en la página~30.

3.2 ESTUDIOS RELACIONADOS

3.2.1 Application of Frontal EEG Asymmetry to User Experience Research

En esta publicación utilizan la tecnología EEG para comprobar la experiencia de usuario de un sitio web, siendo su objetivo investigar la viabilidad de utilizar

la neurofisiología como indice para evaluar la experiencia de usuario.

Para cuantificar los resultados, utiliza la asimetría alfa frontal como índice principal, teniendo en cuenta el modelo de Davidson comentado en la sección 2.4.2 en la página~9, en el que la parte izquierda procesa los efectos positivos, mientras que el derecho procesa los negativos.

Este estudio, como se ha comentado anteriormente, establece que la relación entre la intensidad de las ondas alfa y la activación del cerebro es inversa. Por tanto, nosotros a la hora de mostrar la activación también lo consideraremos así.

3.2.2 A review of brain oscillations in perception of faces and emotional pictures

En este estudio muestran a los sujetos diferentes imágenes emocionales y de expresiones faciales, para recopilan datos de la asimetría de las diferentes ondas cerebrales.

A partir de los datos obtenidos, llegan a la conclusión de que, en comparación con el resto de ondas, la banda de las ondas alfa ofrece unos resultados inconsistentes. Esto choca con el estudio anterior, que se basa en las ondas alfa.

No obstante, aclaran que puede ser debido a los varios métodos matemáticos que han aplicado para el análisis de las frecuencias alfa.

3.3 TECNOLOGÍAS RELACIONADAS

Existen diversas alternativas en cuanto a dispositivos EEG se refiere. Podemos encontrar desde dispositivos profesionales de varios miles de euros, que requieren mucha preparación, condicionando mucho las situaciones en las que pueden ser utilizados; hasta dispositivos de una sorprendente sencillez y que sus electrodos funcionan en seco, permitiendo su utilización en cualquier momento.

Este último tipo de dispositivos es el que nos interesa para este proyecto, ya que uno de sus objetivos es poder obtener datos mientras se realizan actividades cotidianas.

Si observamos el mercado orientado a usuarios finales, encontramos varios dispositivos. Los más destacados son:

Figura 10.: Logo neurosky

Neurosky: El producto que ofrece Neurosky es el headset *MindWave*. Es el más económico, con un precio de 79$ y el que mayor resolución de datos ofrece, con 512 Hz. Aunque es uno de los más limitados al mismo tiempo, ya que únicamente tiene un electrodo, ubicado en la parte frontal.

Figura 11.: Logo MUSE

Muse: La *headband* creada por InterAxon ofrece la lectura de ondas cerebrales a través de cuatro sensores simultáneamente, con una frecuencia de actualización de 256 Hz. Con un precio de 299$, se encuentra en la media por las especificaciones que ofrece.

Figura 12.: Logo emotiv

Emotiv: Con dos headsets a la venta, Emotiv es donde más variedad encontramos, en cuanto a dispositivos propietarios se refiere. Los dispositivos son el EPOC+, con 14 sensores y muestreo a 256 Hz, y el INSIGHT, un headset de 5 canales y 128 Hz. Con un precio de 799 y 299$ respectivamente, se antojan algo caros si se comparan con la competencia.

Figura 13.: Logo OpenBCI

OpenBCI: A diferencia de los dispositivos mencionados anteriormente, OpenBCI no ofrece un headset como tal, si no que ofrece una plataforma *OpenSource* con la que crear un dispositivo a medida, con la cantidad y tipo de sensores que sean necesarios para cada aplicación.

3.4 ELECCIÓN DEL DISPOSITIVO

Ahora que conocemos el estado del mercado y los conocimientos necesarios sobre las ondas cerebrales, podemos analizar las cracterísticas y escoger el dispositivo que mejor se ajuste a nuestras necesidades y presupuesto.

Como hemos visto en la sección 2.2 en la página~6, las ondas cerebrales más representativas se encuentran en el rango de 0-100 Hz. Sabemos, según el Teorema de muestreo de Nyquist-Shannon[2], que si la frecuencia más alta contenida en una señal analógica es F y la señal se muestrea a una tasa >= $\boldsymbol{2F}$ entonces la señal se puede recuperar totalmente a partir de sus muestras.

Por tanto, para obtener unos datos completos, vamos a descartar los dispositivos que su tasa de muestreo sea inferior a 200Hz. Además, queremos comprobar la actividad de estas ondas en el mismo instante de tiempo desde distintos puntos simultáneamente, lo que conlleva descartar aquellos dispositivos que cuenten

únicamente con un sensor.

Llegados a este punto, hay que considerar también el tipo de tecnología EEG que utilizan. Mientras que el dispositivo de Neurosky y la headband de MUSE utilizan electrodos secos, los dispositivos de Emotiv requieren una solución salina en sus electrodos para funcionar. Esto es un problema que limita la movilidad del dispositivo, ya que para su utilización hay que colocar y humedecer cada electrodo uno a uno. Al finalizar su utilización, se deben desmontar y guardar de nuevo. Además, conforme se va secando el producto, la fiabilidad de las mediciones realizadas va disminuyendo. Esto conlleva que no sean aptos para su uso en cualquier lugar.

Tras aplicar estos filtros, el Neurosky Mindwave y los dispositivos de Emotiv quedan descartados. Neurosky Mindwave no es válido para nuestros propósitos por contar con un sólo sensor en la parte frontal. De los dos dispositivos que hemos considerado de Emotiv, uno queda descartado por no llegar a la tasa de muestreo requerida (necesitamos un muestreo de al menos 200 Hz), y el otro por salirse ampliamente del presupuesto. Además, ambos utilizan electrodos humedecidos en una solución salina y estamos buscando un dispositivo portátil que pueda funcionar en seco, por lo que quedan doblemente descartados. Llegados a este punto, los candidatos son dos: MUSE o OpenBCI.

Finalmente el dispositivo elegido para llevar a cabo las pruebas es MUSE debido principalmente a su precio y sencillez de utilización.

Otros motivos por los que ha sido escogido es por que MUSE ofrece un SDK sencillo y multiplataforma, siendo compatible con Android, que es donde va a correr nuestra aplicación.

3.4.1 MUSE Headband

Ahora que hemos decidido qué dispositivo utilizar, vamos a conocer a fondo el dispositivo y los datos que puede proporcionarnos.

3.4.1.1 *Sensores*

La headband MUSE cuenta con cinco **electrodos**. Según la versión web de la documentación, los electrodos con los que cuenta son TP9, Fp1, Fp2, TP10 y el sensor de referencia FPZ, que únicamente detecta contacto, pero no recopila datos. Aunque en la versión más actualizada de la misma, en PDF, consideran que debido al tamaño medio de la cabeza la nomenclatura TP9 y TP10 no es precisa en la mayoría de los casos, y que por lo tanto es más correcto considerar a los electrodos TP9 y TP10 como AF7 y AF8 respectivamente, ya que es la posición más común en la que suelen ser colocados. Como la documentación más actualizada los considera AF7 y AF8, los vamos a considerar de esta forma.

Además de los electrodos, la headband cuenta también con **acelerómetro**, para detectar el movimiento de la misma cuando está colocada en la cabeza del usuario.

Podemos observar la ubicación de los electrodos en la diadema en la siguiente imagen:

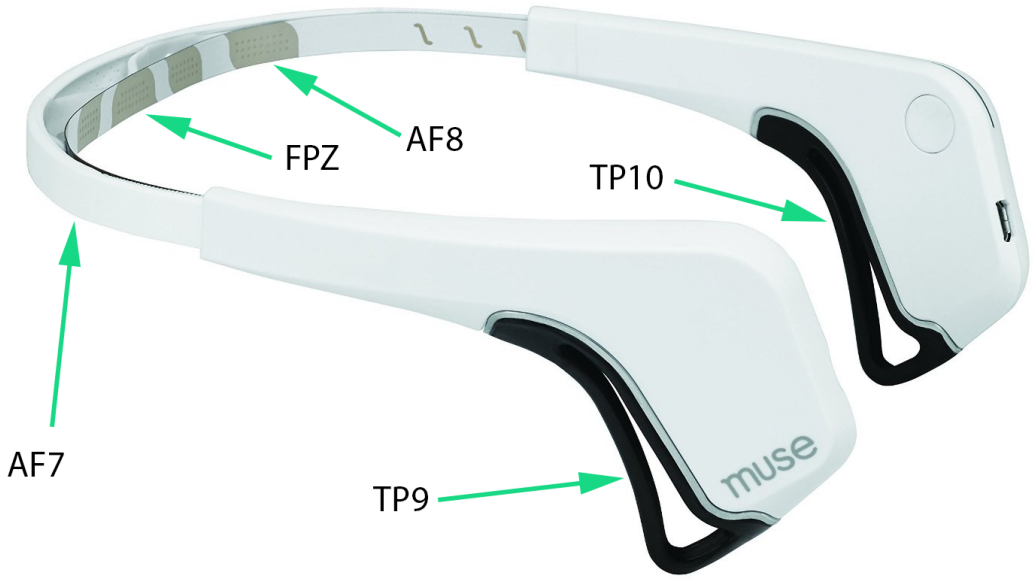


Figura 14.: Ubicación física de los distintos electrodos en la diadema.

3.4.1.2 *SDK*

Las herramientas que ofrece MUSE para desarrolladores son bastante completas y multiplataforma. En el caso de Android, la headband no ofrece algunos datos que en las herramientas de PC sí, como la FFT en tiempo real, por lo que para obtener este resultado, tendremos que realizar los cálculos en la aplicación.

El SDK de Android en concreto, ofrece varias herramientas útiles como la gestión de la conexión Bluetooth entre el dispositivo y la headband. Además, ofrece muchos datos directamente desde la headband sin tener que realizar un procesado del EEG en raw, como por ejemplo, los valores de cada banda de frecuencias, en forma absoluta o relativa; o detección de parpadeo.

4 DESARROLLO

Una vez escogido el sistema con el que obtendremos los datos, la plataforma para la que desarrollar la aplicación, definidos los objetivos y teniendo los conocimientos para poder tomar decisiones informadas, comienza la fase de desarrollo.

4.1 DISEÑO

4.1.1 Análisis de requerimientos

Definiremos los requerimientos funcionales y no funcionales a partir de las necesidades que debe cubrir la aplicación.

4.1.1.1 *Requerimientos funcionales*

- **R1.** El sistema debe ser capaz de obtener los datos del dispositivo MUSE.
- **R2.** El sistema debe permitir visualizar los datos.
- **R3.** El sistema debe permitir escoger qué datos se visualizan
- **R4.** El sistema debe proporcionar la forma de guardar en un fichero los datos obtenidos.
- **R5.** El sistema debe permitir escoger qué datos se guardan
- **R6.** El sistema debe permitir establecer una marca para la sincronización.
- **R7.** El sistema debe permitir el registro e inicio de sesión de usuarios.
- **R8.** El sistema debe permitir a un usuario registrado subir un fichero para ser guardado en el Cloud.
- **R9.** El sistema permitirá compartir un fichero subido al Cloud.

4.1.1.2 *Requerimientos no funcionales*

- **RNF 1.** El dispositivo debe disponer de conexión Bluetooth.
- **RNF 2.** El dispositivo debe ejecutar la versión 4.4
- **RNF 3.** Para que funcione la conexión al Cloud el dispositivo debe disponer de conexión a Internet.

4.1.2 Interfaz Gráfica

La interfaz de la aplicación es una parte importante de la aplicación, ya que es con lo que se encontrará el usuario en el momento de utilizarla. La interfaz de la aplicación tendrá la siguiente estructura:

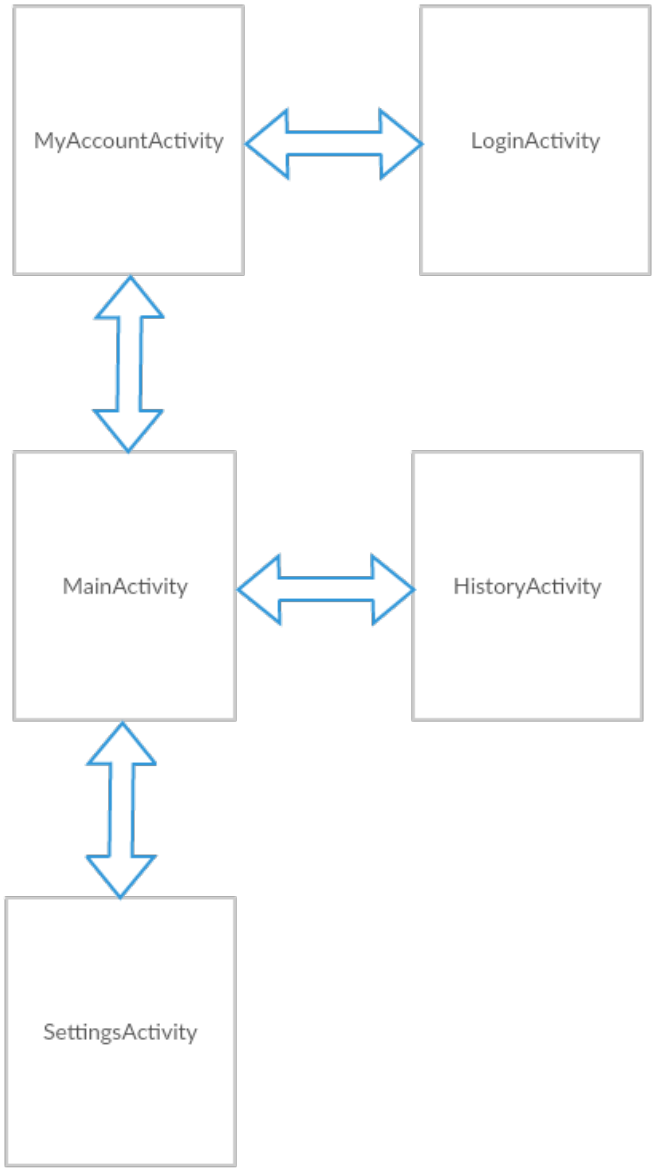


Figura 15.: Esquema de la interfaz de usuario.

A continuación vamos a mostrar el diseño de las pantallas que componen la aplicación y una breve explicación de las mismas.

4.1.2.1 *Pantalla principal*

La pantalla principal tendrá el siguiente diseño:

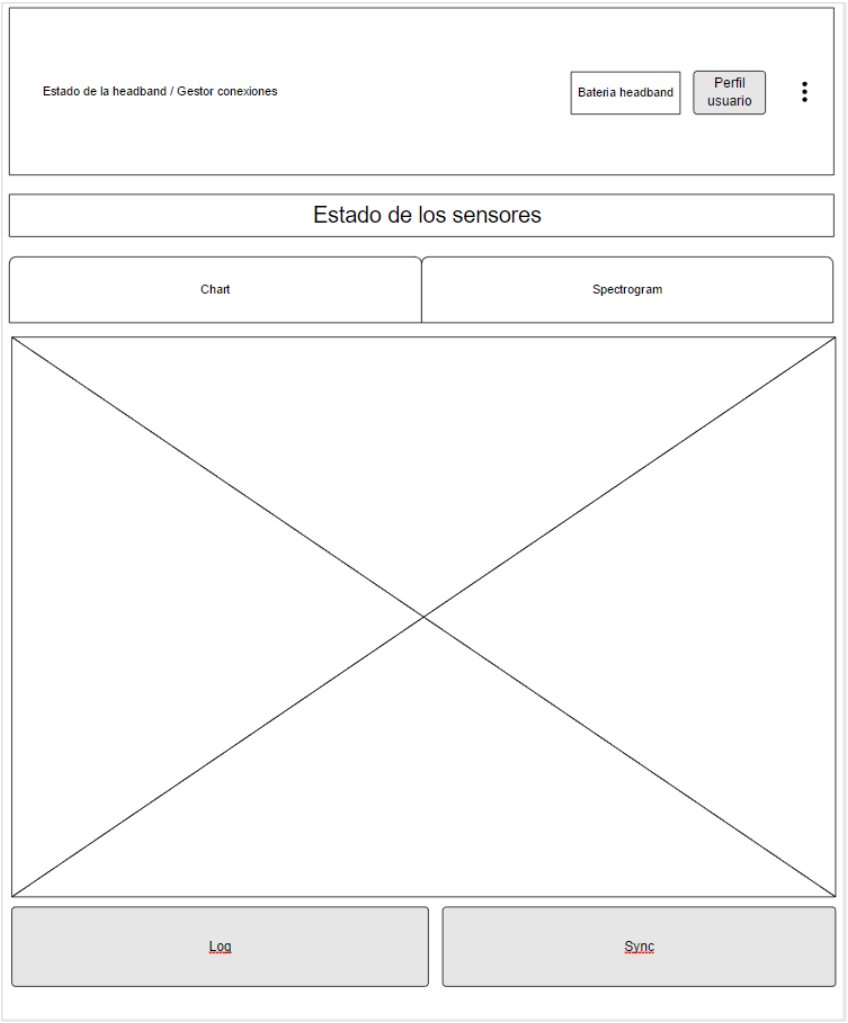


Figura 16.: Diseño Pantalla principal

Esta pantalla está compuesta por varias secciones:

- **Barra de estado:** Contendrá una vista que mostrará el estado de la conexión con la headband, y otra que muestre el estado de la batería de la misma. Además de un botón para acceder al perfil e iniciar sesión, y el botón de menú. Al seleccionar la vista de estado, se mostrará el menú de conexión con la headband si estaba desconectada o finalizará la conexión en caso contrario. Al seleccionar el botón Perfil de usuario se abrirá la pantalla del perfil del usuario. El menú contendrá la opción Historial, que al ser seleccionada, abrirá la pantalla Historial.
- **Estado de los sensores.** Esta vista mostrará una indicación del estado de la conexión de los sensores de la headband.
- **Pestañas *Chart* y *Spectrogram*:** Permitirán cambiar el placeholder entre la visualización de las ondas y un espectrograma.
- **Botones inferiores:** Log iniciará la escritura de los ficheros y Sync generará una marca en los mismos para poder sincronizar las mediciones.

4.1.2.2 *Pantalla Perfil de usuario*

Esta pantalla podrá tener dos aspectos, dependiendo del estado del login:

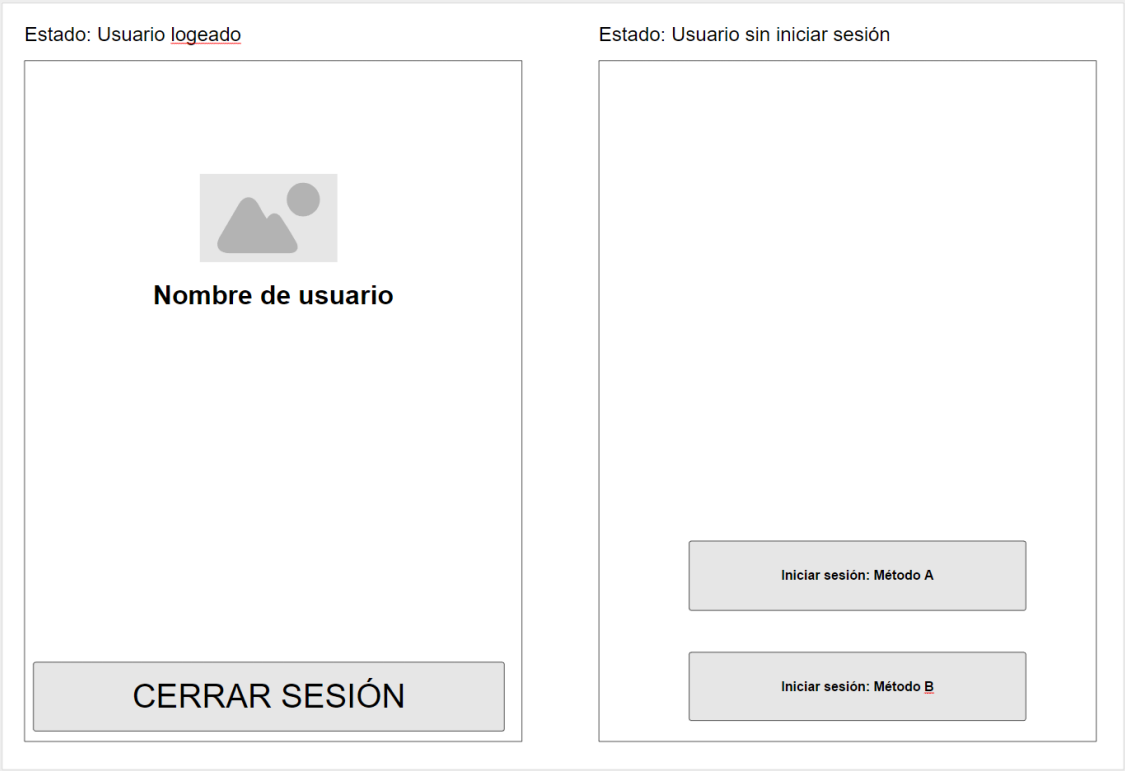


Figura 17.: Pantalla de perfil, en la izquierda cuando hay un usuario con sesión iniciada, en la derecha cuando no lo hay

Cuando el usuario está logeado, mostrará su imagen de perfil, su nombre y un botón, Cerrar sesión que al ser seleccionado, cerrará la sesión del usuario y volverá a la página principal.

4.1.2.3 *Pantalla Historial*

Aquí se mostrarán los distintos ficheros que el usuario haya grabado tras realizar alguna sesión.

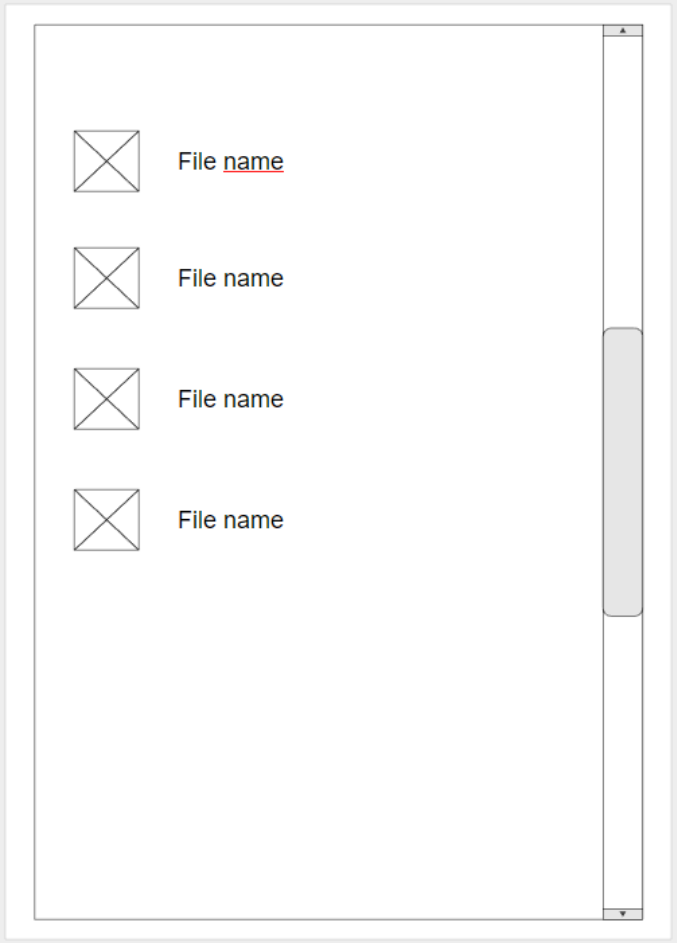


Figura 18.: Diseño Pantalla Historial

La lista de ficheros permitirá subir y bajar ficheros al cloud, además de compartir los ficheros del cloud y de visualizar el contenido de cualquiera de los ficheros al ser seleccionado.

4.1.3 Guardado de datos

Según la documentación del SDK Android de MUSE[3], cada paquete de datos que es recibido por la aplicación contiene el valor del tipo determinado de cada electrodo de la headband. De esta forma, cuando recibimos un paquete RAW EEG, obtenemos el valor de la EEG en cada uno de los cuatro electrodos.

Como ya hemos comentado con anterioridad en la sección 2.4.2 en la página~9, vamos a utilizar la asimetría frontal, por lo que descartaremos los valores de los sensores laterales (TP9 y TP10), y guardaremos los provenientes de los sensores AF7 y AF8.

De esta forma, obtendremos en cada paquete una marca de tiempo de la captura y el valor de la medida solicitada en cada hemisferio.

Los datos que vamos a guardar son de tres tipos:

1. **Raw EEG:** La medida de la EEG sin tratar, tal como la obtiene la headband.
2. **Alpha Absolute:** El valor de las ondas alfa, que lo obtiene la headband y lo ofrece mediante su SDK.
3. **FFT de los datos EEG:** Transformada de los datos Raw EEG.

Una vez hemos establecido los datos que vamos a guardar, podemos escoger el formato y diseñar la estructura que van a tener los distintos ficheros. Como formato de datos vamos a utilizar CSV (Comma Separated Values), que debido a la estructura en columna de los datos, es el más sencillo y es compatible con la amplia mayoría de software de análisis de datos.

Los ficheros de los datos Raw EEG y Alpha Absolute, ya que ambos datos llegan en el mismo formato directamente de la headband, van a tener la misma estructura. La estructura es la siguiente: Timestamp, valorIzquierdo, valorDerecho.

Para el fichero que contendrá el resultado de la FFT, deberemos establecer otra estructura. Esta estructura será la siguiente: Timestamp, Canal correspondiente (izquierdo o derecho), y 128 valores correspondientes a los valores de la FFT.

Podemos ver un ejemplo de cómo resultan finalmente los ficheros una vez implementados siguiendo esta especificación en las figuras 28 y 29 en la página~28

4.2 IMPLEMENTACIÓN

Como fase final del proceso de desarrollo de la aplicación, paso a explicar la implementación de la misma.

4.2.1 Tecnologías utilizadas

4.2.1.1 *Android Studio*

Figura 19.: Logo Android Studio

Desde que fuera presentado en 2013 como una alternativa a Eclipse, Android Studio se ha consolidado como la herramienta de referencia para el desarrollo de aplicaciones Android.

Basado en la versión Open Source del popular IDE IntelliJ IDEA creado por la empresa Jet Brains, ofrece un entorno específico para Android, con todas las herramientas necesarias, además de tener todo el ecosistema de plugins de IntelliJ disponible para aumentar sus funcionalidades.

4.2.1.2 *RxJava*

RxJava es la implementación en Java de las Reactive Extensions, una librería que trata de facilitar la programación asíncrona y basada en eventos mediante el uso de Observables. Está basada en el patrón Observer el cual modifica añadiéndole nuevos operadores, que permiten componer distintos Observables. Permite gestionar de una forma más sencilla el *threading* y la sincronización de eventos.

Figura 20.: Logo Reactive Extensions

RxJava es una gran alternativa a otras formas de trabajar en background comúnmente utilizadas en Android como son las AsyncTask, pero sin los problemas de estas últimas, como memory leaks. Además está pensado para trabajar con Streams de datos, que es exactamente lo que recibimos desde la headband.

4.2.1.3 *LibMuse*

LibMuse es la implementación para Android del SDK de MUSE. Permite la comunicación con la diadema y su control desde el software cliente. Aporta las herramientas para poder obtener las mediciones necesarias para nuestros propósitos.

4.2.1.4 *Firebase*

Figura 21.: Logo Firebase

Firebase es lo que se conoce como un MBaaS (Mobile Backend as a Service), que ofrece multitud de herramientas para aplicaciones móviles. Permite disponer de un backend sin la necesidad de construirlo nosotros mismos, todo esto sin pagar, ya que es gratuito mientras no sobrepasemos las cuotas de uso gratuito (Que son bastante altas).

Ofrece multitud de herramientas para facilitar el desarrollo de aplicaciones móviles, como una base de datos en tiempo real, autenticación de usuarios, notificaciones push... En nuestro caso, utilizaremos únicamente la Autenticación de usuarios y la parte de Almacenamiento, para poder hacer copias de seguridad/-guardar/compartir fácilmente los ficheros que obtengamos de las mediciones.

4.2.2 Creación del proyecto

Debido a las dimensiones del proyecto, se han organizado las diferentes clases que lo integran en diferentes paquetes, en función de su rol en la aplicación. Esta organización semántica permite que a la hora de buscar un elemento en concreto, sea posible reconocer de un vistazo dónde se encuentra. Esto a la larga, que en un proyecto pequeño sería un tiempo despreciable, en uno de grandes di-

mensiones se convierte en un ahorro de tiempo considerable.

Estos paquetes son:

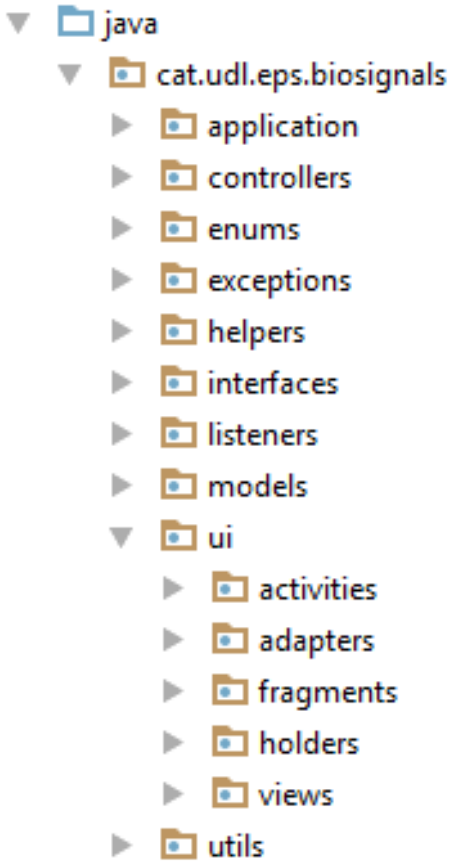


Figura 22.: Estructura del proyecto Android

4.2.3 Recepción de datos

La recepción de los datos es gestionada por el SDK de Muse para Android, Libmuse. Este SDK ofrece un proyecto de ejemplo a partir del cual comenzar la integración.

Para poder recibir datos, en primer lugar hay que tener la headband conectada mediante Bluetooth al dispositivo Android. Una vez está conectada, el SDK ya puede acceder a ella.

Para poder utilizar el SDK lo primero que debemos hacer es declarar un objeto *MuseManagerAndroid*. Una vez declarado, lo inicializamos y establecemos el *Context*.

```
// Declaración
private MuseManagerAndroid manager;
// Inicialización
manager = MuseManagerAndroid.getInstance();
manager.setContext(this);
```

Figura 23.: Inicialización del SDK de Muse

Tras ser inicializado, ya podemos obtener la lista de *Muses* disponibles en el dispositivos. Para ello, hay que añadir un *MuseListener* al manager. De forma que cuando el manager detecte una headband se actualice la lista.

```
// Lista de dispositivos muse
private List<Muse> museList;

// Listener de cambios.
final MuseListener museListener = new MuseListener() {
    @Override
    public void museListChanged() {
        museList = manager.getMuses();
    }
};

// Añadimos el listener al manager
manager.setMuseListener(museListener);
```

Figura 24.: Añadimos el listener al manager para descubrir las headbands sincronizadas con el dispositivo Android.

Una vez disponemos de la lista de dispositivos a los que podemos conectarnos, podemos proceder a iniciar la recepción de datos. Para recibir los datos, debemos registrar dos listeners con la instancia de la headband a la que queremos conectarnos. Estos listeners son un MuseConnectionListener y MuseDataListener, encargados de recibir notificaciones sobre el estado de la conexión y de recibir los datos respectivamente.

Debido a la estructura de Libmuse, en cada ocasión en la que necesitásemos obtener acceso al stream de datos deberíamos repetir los pasos anteriores, además de crear y registrar un listener específico para cada caso de uso. Para evitar esto y poder reutilizar el mismo listener pudiendo modificar su comportamiento, se ha utilizado RxJava ya que permite centralizar la recepción de los datos. Así, el acceso a los datos puede ser realizado con una sola llamada, en lugar de tener que generar varias clases nuevas por cada lugar en el que se quiera acceder a ellos.

El registro de los listeners con la instancia de la headband ocurre cuando nos suscribimos a un stream de datos. Esta suscripción la realizamos mediante RxJava. Además, la conexión con Muse la gestiona un Singleton. De esta forma, podemos acceder a los datos desde cualquier parte de la aplicación sin tener que volver a realizar toda la conexión de nuevo. Esto es posible debido a la implementación de MuseDataListener que estamos empleando. Esta implementación está detallada más adelante, en la sección 4.2.3.1 en la página siguiente

Para acceder a los streams de datos, el singleton ofrece el siguiente método:

```
public Observable<MuseDataPacket> observeMuseData(final Muse muse) {
    return Observable.create(emitter -> {
        if (museManager != null) {
            museDataListener.addMuseDataPacketSubscriber(emitter);
            muse.registerDataListener(museDataListener, MuseDataPacketType.BATTERY);
            muse.registerDataListener(museDataListener, MuseDataPacketType.QUANTIZATION);
            muse.registerDataListener(museDataListener, MuseDataPacketType.HSI_PRECISION);
            muse.registerDataListener(museDataListener, MuseDataPacketType.EEG);
            muse.registerDataListener(museDataListener, MuseDataPacketType.ALPHA_ABSOLUTE);
            muse.registerDataListener(museDataListener, MuseDataPacketType.BETA_ABSOLUTE);
            muse.registerDataListener(museDataListener, MuseDataPacketType.DELTA_ABSOLUTE);
            muse.registerDataListener(museDataListener, MuseDataPacketType.THETA_ABSOLUTE);
            muse.registerDataListener(museDataListener, MuseDataPacketType.GAMMA_ABSOLUTE);
        } else {
            emitter.onError(new Throwable(THROWABLE_NULL_MANAGER));
            muse.unregisterAllListeners();
        }
    });
}
```

Figura 25.: Creamos el observable al cual le pasamos los eventos que debe escuchar, y registramos el listener.

Una vez creado el observable debemos observar los cambios para obtener los datos:

```
private void listenEEG(final Muse muse) {
    museDataSubscription = MuseHelper.getInstance(this)
            .observeMuseData(muse)
            .subscribeOn(Schedulers.io())
            // Si el paquete es del tipo EEG, lo procesará
            .filter(museDataPacket -> museDataPacket.packetType() == MuseDataPacketType.EEG))
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(museDataPacket -> {
                // Procesar paquete recibido
            });
}
```

Figura 26.: Nos suscribimos al stream de datos y filtramos los que nos interesa procesar en este caso.

4.2.3.1 *MuseDataListener*

La implementación que se ha utilizado de MuseDataListener permite mantener varias suscripciones activas al mismo tiempo y de esta forma obtener los datos en diferentes partes de la aplicación.

Esto se consigue gracias a la librería RxJava, que ofrece las herramientas necesarias. Como vemos en la figura XX, cuando se llama al método observeMuseData, añadimos a una lista al emisor de esta llamada de la misma forma que añadimos un observador en el patrón Observer.

Así, cuando el listener recibe un paquete de datos, itera sobre la lista de emisores informándoles del paquete que acaba de recibir. Una vez han sido informados, el observador es quien decide si el paquete le interesa o no.

El listener tiene la siguiente estructura:

```
public class MuseDataListenerImpl extends MuseDataListener {

    private List<ObservableEmitter<MuseDataPacket>> museDataPacketSubscribers;
    private List<ObservableEmitter<MuseArtifactPacket>> museArtifactPacketSubscribers;

    @Override
    public void receiveMuseDataPacket(MuseDataPacket museDataPacket, Muse muse) {
        if (museDataPacketSubscribers != null) {
            for (ObservableEmitter<MuseDataPacket> emitter : museDataPacketSubscribers) {
                if (!emitter.isDisposed()) emitter.onNext(museDataPacket);
                else {
                    museDataPacketSubscribers.remove(emitter);
                }
            }
        }
    }

    @Override
    public void receiveMuseArtifactPacket(MuseArtifactPacket museArtifactPacket, Muse muse) {
        if (museArtifactPacketSubscribers != null) {
            for (ObservableEmitter<MuseArtifactPacket> emitter : museArtifactPacketSubscribers) {
                if (!emitter.isDisposed()) emitter.onNext(museArtifactPacket);
                else museArtifactPacketSubscribers.remove(emitter);
            }
        }
    }

    public void addMuseDataPacketSubscriber(ObservableEmitter<MuseDataPacket> museDataPacketSubscriber) {
        if (museDataPacketSubscribers == null) museDataPacketSubscribers = new ArrayList<>();

        museDataPacketSubscribers.add(museDataPacketSubscriber);
    }

    public void addMuseArtifactPacketSubscriber(ObservableEmitter<MuseArtifactPacket> museArtifactPacketSubscriber) {
        if (museArtifactPacketSubscribers == null) {
            museArtifactPacketSubscribers = new ArrayList<>();
        }
        museArtifactPacketSubscribers.add(museArtifactPacketSubscriber);
    }
}
```

Figura 27.: Implementación del listener, que notifica a los observers suscritos al evento.

4.2.4 Guardado de datos

En cuanto al guardado de los datos en ficheros, tras realizar algunas pruebas, vamos a utilizar una variante del formato CSV. En un principio se había utilizado el formato CSV original, pero ciertos programas de análisis de datos (MS Office Excel, LibreOffice Calc), al encontrarse con números decimales y comas en el mismo documento no detectan bien las separaciones e importan mal el documento, lo que imposibilita su utilización. El problema surge de que en ciertas configuraciones regionales los decimales se dividen con comas y en otras con puntos. Este comportamiento es posible cambiarlo desde la configuración de los programas, pero para evitar tener que dar pasos adicionales a la hora de analizar los datos, cambiaremos el separador utilizado a uno que funcione correctamente en cualquier caso, sin cambiar la configuración.

Para solventar este problema se ha sustituido el separador actual,(,) comas, por el símbolo (;) punto y coma. Esto permite mantener la estructura del fichero intacta y evita los problemas mencionados anteriormente.

En cuanto a la implementación, se ha utilizado el patrón Singleton de forma que resulta muy sencillo de utilizar. La escritura de los datos ocurre en un thread en background para no sobrecargar la UI, pues es una tarea intensiva. Además,

debido al gran tamaño que pueden llegar a tener los ficheros, desde los ajustes de la aplicación se puede establecer qué ficheros de datos guardar, a elección del usuario.

La clase encargada de gestionar el guardado de los datos es **FileProcessor**. La API que expone varios métodos públicos para gestionar la escritura de los datos. Estos métodos son:

- **boolean** initLogger(*String title, SharedPreferences prefs*): Inicializa el singleton y prepara los diferentes FileWriters para escribir los diferentes ficheros seleccionados. Si se inicializa correctamente, retorna true, en caso de error false.
- **boolean** stopLogging(): Cierra los FileWriters y los ficheros. Retorna true si todo ha ido correcto, false en caso de error.
- **void** write(*MuseDataPacket museDataPacket*): Escribe los datos recibidos del headset en el formato de EEG especificado en la sección 4.1.3 en la página~20. Los datos son guardados en el fichero correspondiende, ALPHA_WAVES o RAW_EEG, dependiendo del tipo de paquete que llega.
- **void** write(*double[] leftFFT, double[] rightFFT*): Recibe el resultado del cálculo de la FFT y lo escribe en el fichero.
- **void** createMarkOnFiles(): Este método es utilizado para la sincronización y como su nombre indica, crea una marca en los ficheros. La marca sigue el formato del tipo de fichero en el que se realiza. En los ficheros EEG, el contenido es Timestamp;SYNC;MARK. En cambio, en el archivo FFT al tener un formato distinto, su contenido es Timestamp;SYNC_MARK; y el resto de campos (128) rellenados con ceros.

El resultado de una sesión de grabación de datos, con el guardado de todos los tipos de datos activado y con el nuevo formato, utilizando el nuevo separador lo podemos ver a continuación:

1496576064743440	LEFT_CHANNEL	-114.215	175.4247	-105.057	-110.314	233.7551	-280.752	209.2599	-94.6109	23.95606	-21.8
1496576064744190	RIGHT_CHANNEL	49.93316	-131.102	189.4967	7.777174	-243.457	257.7871	-186.816	85.84379	48.38723	-56.7
1496576064744940	LEFT_CHANNEL	-225.581	346.5596	-336.944	242.401	-276.74	298.6975	-246.532	157.211	-97.5673	121.0
1496576064745690	RIGHT_CHANNEL	282.4759	-420.364	372.2592	-259.27	323.7422	-295.859	221.0777	-130.667	39.4991	-108.
1496576064746440	LEFT_CHANNEL	-288.536	371.5114	-255.767	100.2935	85.09905	-219.547	254.8931	-171.651	43.17877	69.40
1496576064747190	RIGHT_CHANNEL	438.552	-530.693	235.3255	28.22897	-125.37	219.857	-225.202	157.0807	-74.9957	-2
1496576064747940	LEFT_CHANNEL	-258.215	255.4903	29.31015	-186.673	111.5798	98.3104	-223.683	149.5866	19.90466	-55.8
1496576064748690	RIGHT_CHANNEL	362.3256	-400.979	45.04866	236.5488	-143.505	-94.3109	194.766	-140.59	36.01684	26.48
1496576064749440	LEFT_CHANNEL	-127.824	82.1604	139.9837	-89.4726	-142.393	7.392878	176.417	-97.2178	9.785659	-50.
1496576064750190	RIGHT_CHANNEL	212.7536	-201.691	10.85629	-75.9808	187.3052	-7.66941	-140.202	86.592	8.762664	50.73
1496576064750940	LEFT_CHANNEL	25.86105	-51.9139	12.87714	48.28361	89.74191	-45.3718	-123.406	32.20545	-39.3886	-11.
1496576064751690	RIGHT_CHANNEL	70.30371	-34.633	10.35227	-114.008	-39.7459	51.13789	79.1484	-21.8466	-22.719	60.32
1496576064752440	SYNC_MARK	0	0	0	0	0	0	0	0	0	
1496576064752440	SYNC_MARK	0	0	0	0	0	0	0	0	0	
1496576064762190	LEFT_CHANNEL	176.978	-158.901	-69.7755	107.8136	-30.9792	25.87213	82.0773	32.38943	-18.635	-39.2
1496576064761440	RIGHT_CHANNEL	-153.829	155.3231	-52.6155	99.46433	-37.4686	-46.9921	-29.1869	-39.5107	-2.53889	-9.74
1496576064760690	LEFT_CHANNEL	149.5877	-170.424	50.07097	42.85806	-32.7763	12.07649	-57.5502	-53.9073	112.8911	6.137
1496576064759940	RIGHT_CHANNEL	-204.756	232.5329	-96.6907	32.09057	-22.1662	19.0859	-4.60673	79.90652	-9.88044	-102.
1496576064759190	LEFT_CHANNEL	55.55475	-124.342	186.7044	-120.868	50.93587	-35.9982	52.14472	10.4568	-87.8107	121.9
1496576064758440	RIGHT_CHANNEL	-172.699	237.7452	-99.1933	-110.949	50.07666	4.744377	11.20338	-82.6114	85.22238	-35.7
1496576064757690	LEFT_CHANNEL	35.49297	-64.0846	108.0967	-94.0678	-13.0801	41.76306	-50.588	68.65473	-65.1082	47.57
1496576064756940	RIGHT_CHANNEL	-147.137	192.9277	-119.689	12.83632	33.42781	-21.136	-4.50688	29.28728	-78.4215	109.5
1496576064756190	LEFT_CHANNEL	9.024995	27.99124	-113.617	87.64179	7.251214	-25.6212	43.53544	-110.201	143.4406	-118.

Figura 28.: Contenido del fichero de datos FFT

1496576064746280	820.8452	855.3898
1496576064746790	824.1352	857.0348
1496576064747260	822.4902	855.3898
1496576064747740	820.8452	853.7448
1496576064748210	822.4902	855.3898
1496576064748690	822.4902	852.0998
1496576064749190	815.9102	855.3898
1496576064749640	819.2002	850.4548
1496576064750080	817.5552	850.4548
1496576064750540	819.2002	852.0998
1496576064751000	820.8452	848.8099
1496576064751480	825.7801	850.4548
1496576064751960	817.5552	853.7448
1496576064752440	SYNC	MARK
1496576064832840	820.8452	852.0998
1496576064833990	824.1352	853.7448
1496576064834320	820.8452	855.3898
1496576064834560	822.4902	853.7448
1496576064834770	820.8452	853.7448
1496576064834980	822.4902	853.7448
1496576064835170	825.7801	853.7448
1496576064835360	827.4251	853.7448

(a) Datos Raw EEG

Timestamp	leftChann	rightChannel
1496576063377370	0.191432	0.221184
1496576063436950	0.191432	0.221184
1496576063529200	0.191432	0.223861
1496576063660020	0.191432	0.227884
1496576063746310	0.191432	0.232852
1496576063819590	0.191432	0.237587
1496576063962910	0.191432	0.23959
1496576064029970	0.207318	0.246641
1496576064346340	0.244203	0.265492
1496576064394840	0.264939	0.265492
1496576064406290	0.274129	0.265492
1496576064466370	0.27319	0.241734
1496576064535760	0.233035	0.217343
1496576064626600	0.214256	0.209631
1496576064724680	0.206094	0.209631
1496576064752440	SYNC	MARK
1496576064847710	0.206094	0.209631
1496576064970440	0.208789	0.209631
1496576065079720	0.244144	0.209631
1496576065163590	0.240537	0.209631
1496576065265370	0.236886	0.209631

(b) Datos de la onda alfa

Figura 29.: Contenido de los ficheros RAW_EEG y ALPHA_WAVES

Podemos comprobar también que la marca para la sincronización está funcionando correctamente, pues se ha añadido en los tres ficheros con el mismo *timestamp*. Esto nos permitirá más adelante poder relacionar los datos obtenidos con la aplicación, con los datos obtenidos mediante otro tipo de sensores.

4.2.5 Procesado de datos

En el procesado de los datos es dónde se concentra el músculo de la app. Se encarga de aplicar a los datos EEG en crudo recibidos desde la *headband* la Transformada de Fourier discreta. Para facilitar su cálculo, utilizamos la librería JTransforms. La elección de esta librería se debe a la sencillez que ofrece su API al estar escrita en Java, además de por los benchmarks[4] en los que se observa que, para la cantidad de datos que vamos a manejar, es igual de eficiente que una de las librerías de cálculo FFT más utilizadas, FFTW, escrita en C; por lo que no merece la pena adaptar la aplicación para poder ejecutar código en C.

Aplicando el algoritmo FFT, conseguimos convertir la señal del dominio del tiempo al dominio de la frecuencia. En el dominio del tiempo la señal no nos aportaba información que se pudiera analizar, pero una vez aplicado Fourier, pasamos a conocer el valor de cada frecuencia en ese periodo de tiempo, lo que permite distinguir los distintos tipos de ondas generados.

Esto es, cada valor que nos llega desde la *headband* es un sólo número, que representa el valor potencial eléctrico del EEG en ese instante de tiempo, pero lo verdaderamente interesante son las frecuencias que componen ese valor EEG. Ahí es donde entra Fourier, permitiendo transponer los datos de un dominio a otro. Veámoslo con un ejemplo gráfico.

Para mostrar la utilidad de la FFT, vamos a aplicarla a la siguiente muestra de datos EEG:

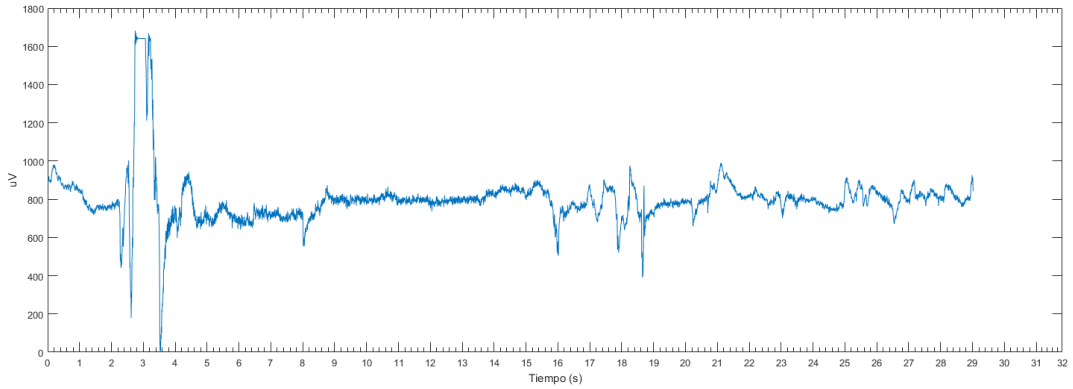


Figura 30.: Datos de una grabación EEG

Escogemos un intervalo de 256 valores y aplicando la Transformada de Fourier sobre esta muestra, obtenemos:

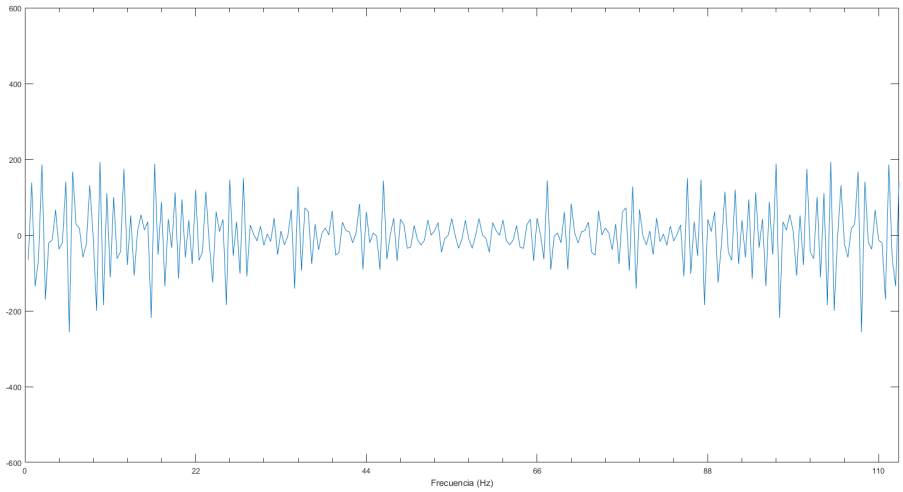


Figura 31.: Aplicando FFT sobre una muestra

Para obtener este resultado, utilizo los mismos parámetros que utiliza MUSE en su SDK de PC[5], ya que como he mencionado anteriormente en la seccion 3.4.1.2 en la página~15, el SDK de Android no ofrece estos valores. De esta forma obtendremos unos valores lo más cercanos posibles a los que obtiene la headband, sin necesidad de utilizar sus herramientas de PC.
Estos valores son:

- FFT cada 256 samples
- Overlap del 90 %, de los 256 samples que se procesan cada vez, solo 25 serán nuevos, el resto son valores anteriores.
- Utilización de la Hamming Window

Pero lo más interesante de la FFT a la hora de visualizar los datos, es que ofrece la posibilidad de visualizar el espectro de la señal, mediante un espectrograma.

4.2.5.1 *Hamming Window*

La Ventana Hamming es una función matemática, propuesta por Richard W. Hamming y tiene la siguiente forma:

$$w(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right)$$

$$\text{Donde} \quad \alpha = \frac{25}{46} \quad \text{y} \quad \beta = \frac{21}{46}$$

Esto se aplica para cada elemento del conjunto de datos al que se le aplica la función, multiplicando su valor por el resultado de la función de ventana, siendo n el índice del elemento.

Mi implementación de la ventana Hamming en la aplicación es la siguiente:

```java
public class HammingWindow {

    private static final double alpha = 25.0 / 46.0;
    private static final double beta = 21.0 / 46.0;

    public static double[] apply(double[] input) {
        return apply(input, 0);
    }

    public static double[] apply(double[] input, int offset) {
        for (int i = offset; i < input.length; i++) input[i] *= value(input.length, i);
        return input;
    }

    private static double value(int length, int index) {
        return alpha - beta * Math.cos(((2 * Math.PI) * index) / (length - 1));
    }
}
```

Figura 32.: Implementación hamming window en Java

4.2.6 Integración con Firebase

En este momento, tenemos los datos procesados y guardados en el dispositivo, pero nos falta poder compartir fácilmente los ficheros guardados. En este punto aprovecharemos la potencia de Firebase para ofrecer un espacio de almacenamiento en Cloud a los usuarios. De esta forma permitimos liberar espacio de los dispositivos o realizar copias de seguridad de los datos más importantes.

4.2.6.1 *Creación del proyecto*

Para empezar, crearemos un proyecto en Firebase. Esto es posible realizarlo desde el Asistente integrado en Android Studio, ubicado bajo el menú Tools ->Firebase. Al entrar en este menú se nos muestra el asistente, que tiene el siguiente aspecto:

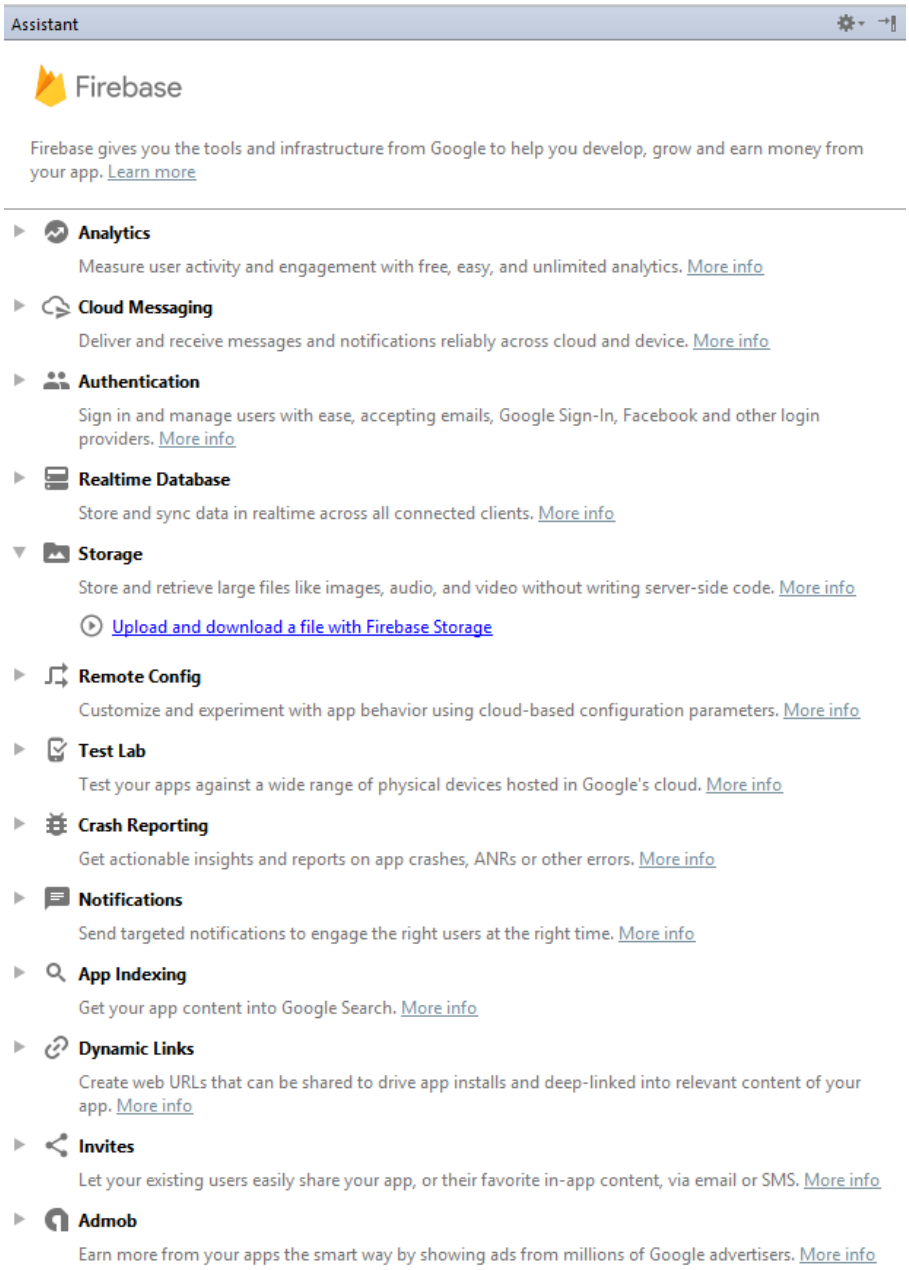


Figura 33.: Pantalla principal del asistente de Firebase

Como podemos ver, nos muestra cada una de las herramientas que ofrece Firebase. En nuestro caso queremos ofrecer almacenamiento, por lo que vamos a hacer click en enlace que vemos en la figura 33 "Upload and download a file with Firebase Storage".

Tras realizar el click, avanzaremos a la siguiente pantalla, en la que como podemos ver en la figura 34 en la página siguiente nos ofrece dos botones, uno para conectar la aplicación a Firebase y otro para añadir la funcionalidad de Storage.

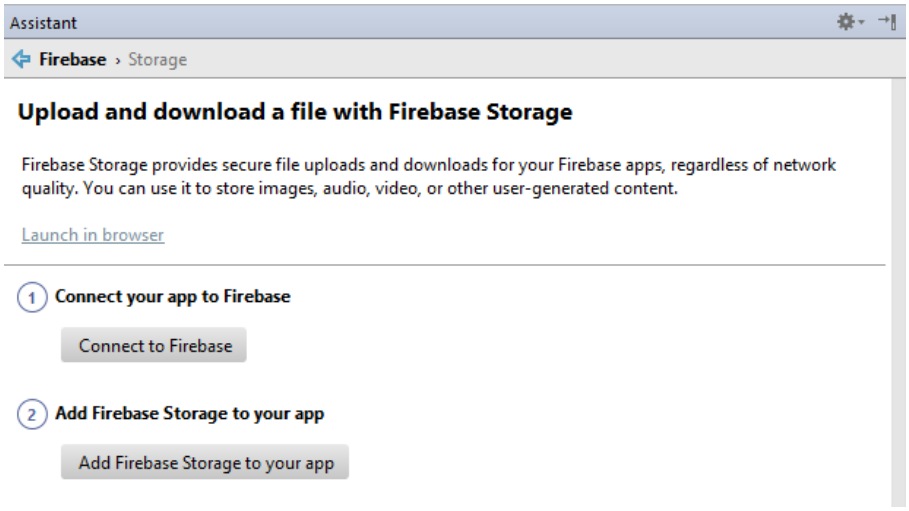


Figura 34.: Detalle de Storage del Asistente

Como todavía no está conectada a Firebase, escogeremos la primera opción, lo cual abre una nueva ventana en la cual nos permite crear el proyecto en Firebase. Escribiremos el nombre que queremos darle al proyecto en Firebase y pulsaremos el botón "Connect to Firebase". Una vez finalizada la creación del proyecto, añadiremos la funcionalidad Storage, que lo que hace es añadir las dependencias al fichero build.gradle de la aplicación. Tras llevar a cabo estas operaciones, estaremos en condiciones de comenzar a utilizar Firebase en la aplicación, y el asistente habrá cambiado del estado que teníamos en la figura 34 al siguiente:

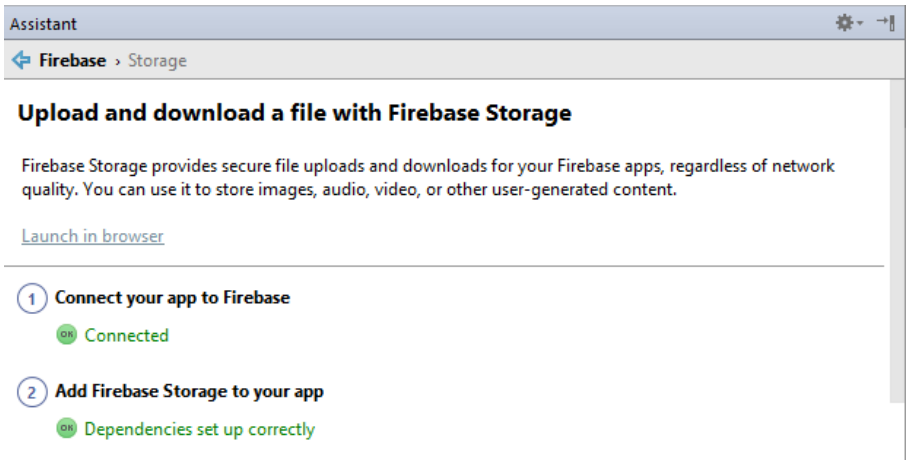


Figura 35.: Asistente tras conectar a Storage

Además, se ha añadido un nuevo archivo al proyecto, llamado google-services.json. Este archivo es el que identifica nuestra aplicación frente a los servidores de Firebase, por lo que es muy importante no modificar su contenido.

4.2.6.2 *Autenticación de usuarios*

Antes de poder permitir a un usuario subir archivos al Cloud hemos de permitir que inicie sesión para poder ofrecer cierto nivel de seguridad sobre los

archivos. En caso de no ofrecer esta funcionalidad, todos los archivos serían públicos.

Para integrar fácilmente esta función, vamos a utilizar la librería FirebaseUI-Android[6], que ofrece una forma rápida y testeada de conectar esta funcionalidad con Firebase.

La librería nos ofrece una implementación sencilla para integrar el inicio de sesión con email y contraseña, y con distintos proveedores, como Google. Para que esta integración funcione, debemos activar los diferentes métodos de inicio de sesión que queremos en la consola del proyecto en Firebase.

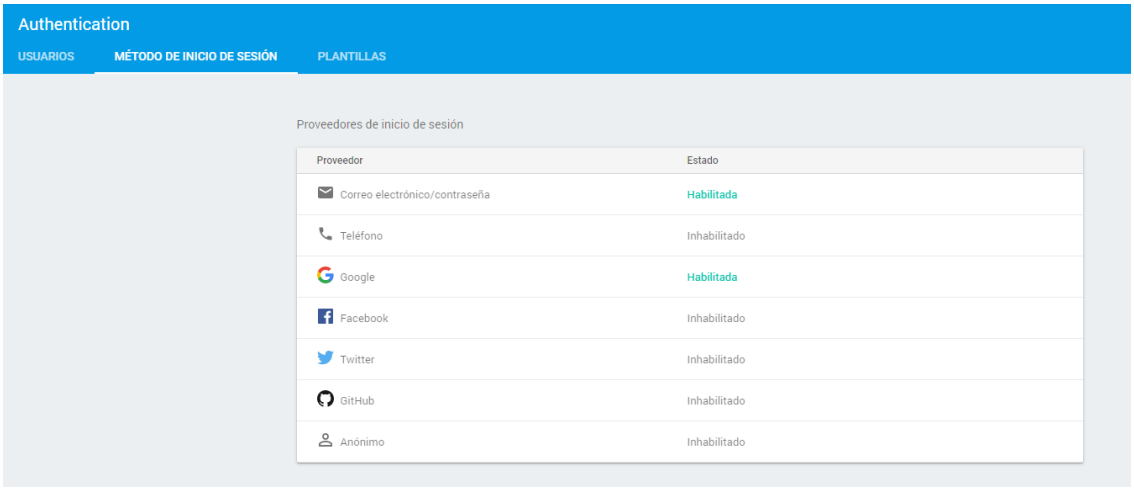


Figura 36.: Métodos de inicio de sesión disponibles

Así, para nuestra aplicación dispondremos de dos métodos de inicio de sesión: Mediante email y contraseña, o utilizando una cuenta de Google.

Una vez activados los métodos de inicio de sesión en la consola, podemos pasar a implementar la autenticación en la aplicación. Esto, debido al uso de la librería, se resume en iniciar la activity correspondiente al login que nos ofrece la librería configurando las opciones pertinentes, como los métodos de inicio de sesión.

Para iniciar la activity de Login y configurar los métodos de inicio de sesión que hemos activado, utilizamos el siguiente código:

```
startActivityForResult(
            AuthUI.getInstance()
                    .createSignInIntentBuilder()
                    .setProviders(Arrays.asList(
                            new AuthUI.IdpConfig.Builder(AuthUI.EMAIL_PROVIDER).build(),
                            new AuthUI.IdpConfig.Builder(AuthUI.GOOGLE_PROVIDER).build()))
                    .setTheme(R.style.AppTheme)
                    .build(),
            RC_SIGN_IN);
```

Figura 37.: Código para lanzar el inicio de sesión

4.2.6.3 *Guardado de los datos en el Cloud*

En este momento, con Firebase conectado a la aplicación, las funciones de Storage añadidas, y solventado el problema de la autenticación, podemos permitir la subida de archivos. La encargada de gestionar la subida de archivos es la clase FirebaseController.

Esta clase ofrece tres métodos principalmente, para realizar las funciones básicas de gestión de archivos: subir, eliminar y descargar archivos.
Ofrece también varios métodos para realizar algunas tareas como obtener el enlace para compartir un fichero ya subido, listar los todos ficheros que ha subido un usuario o saber si se ha iniciado sesión.

Los métodos más importantes son los de gestión de archivos, y son:

- **Task<Void>** performUpload(*Context context, String filePath*): Es la encargada de subir el archivo al cloud. Recibe por parámetro la ruta del fichero, para establecer en qué carpeta guardarlo. Retorna una Task vacía.
- **Task<Void>** performDelete(*Context context, String filePath*): Borra el objeto especificado por parámetro. El resultado de la operación es una Task vacía si todo ha ido correctamente.
- **Task<File>** performDownload(*Context context, String filePath*): Descarga el archivo al almacenamiento local. El resultado de esta operación es una Task que contiene un objeto File, que representa el fichero descargado.

Estos métodos retornan todos una Task, un objeto de la Tasks API, ya que de esta forma las operaciones se llevan a cabo en segundo plano de forma asíncrona y transparente para el usuario.

Tasks API:

Esta API[7] ofrece un conjunto de herramientas orientadas a trabajar de forma asíncrona. Así, podemos dejar que ocurran los eventos que pueden prolongarse mucho en el tiempo y que no sabemos cuando acabarán, y que una vez acabada la tarea nos notifique para continuar y si es preciso tratar el resultado obtenido.

A continuación podemos ver un ejemplo de uso de una Task:

```
public Task<String> getString(int delay) {
    final TaskCompletionSource<String> tcs = new TaskCompletionSource<>();

    new Handler().postDelayed(new Runnable() {
        @Override public void run() {
            tcs.setResult("Resultado");
        }
    }, delay * 1000);

    return tcs.getTask();
}
```

Figura 38.: Tasks API: Declaración de una Task

```
getString(5).addOnSuccessListener(new OnSuccessListener<String>() {
    @Override public void onSuccess(String s) {
        Log.d("Task", "El resultado de la Task es: " + s);
    }
});
```

Figura 39.: Tasks API: Uso de una Task

Mediante esta Task, obtenemos una String con el delay especificado. Durante el tiempo que espera el resultado, la aplicación puede continuar trabajando sin bloquearse y cuando finaliza la ejecución de la tarea se notifica al listener que hemos añadido. Al notificar al listener se ejecuta el código del método onSuccess. En este método se encuentra el resultado obtenido de la Task. De esta forma hemos podido obtener un resultado de forma asíncrona y tratarlo en cuanto lo hemos recibido, sin esperas ni bloqueos.

4.2.7 Interfaz Gráfica

La interfaz de usuario es la última parte que se ha desarrollado de la aplicación, una vez ya funcionaban el resto de partes de la misma, ya que para el correcto funcionamiento del resto de funcionalidades, una interfaz cuidada no era necesaria.

La representación de las ondas se ha implementado utilizando la librería de gráficos MPAndroidChart [8]. De esta forma podemos ver en tiempo real la evolución de las ondas que escojamos.

Para representar la asimetría en la actividad cerebral de forma sencilla, utilizaremos una vista que vaya marcando el lado más activo. Tras buscar alguna forma de representar esto gráficamente de una forma sencilla, no se ha encontrado ninguna librería que ofreciera la funcionalidad requerida para representar los datos de la actividad en cada hemisferio de forma satisfactoria. De esta necesidad nace **PowerBalanceView**, de la que se detalla su implementación más adelante en la sección 4.2.7.2 en la página~39.

La vista del historial de archivos se apoya fundamentalmente en la librería AndroidTreeView[9], que facilita la creación de una estructura arborescente, que encaja perfectamente para representar el tipo de datos que tenemos.

Otro de los elementos importantes es la vista FrequencyView[10], la cual muestra el espectrograma a partir de los datos obtenidos mediante la FFT. Esta vista es parte de de una aplicación desarrollada para mostrar el espectro de una onda de audio. Gracias a que está liberada bajo licencia apache 2.0 y la disposición de su creador, ha sido adaptada para poder representar lo mas fielmente posible la onda EEG.

En cuanto a la internacionalización, la aplicación está adaptada totalmente a dos idiomas: Inglés y Español.

4.2.7.1 *Interfaz de usuario final*

Siguiendo la especificación de la interfaz realizada en la fase de Diseño, el resultado de las diferentes pantallas es el siguiente:

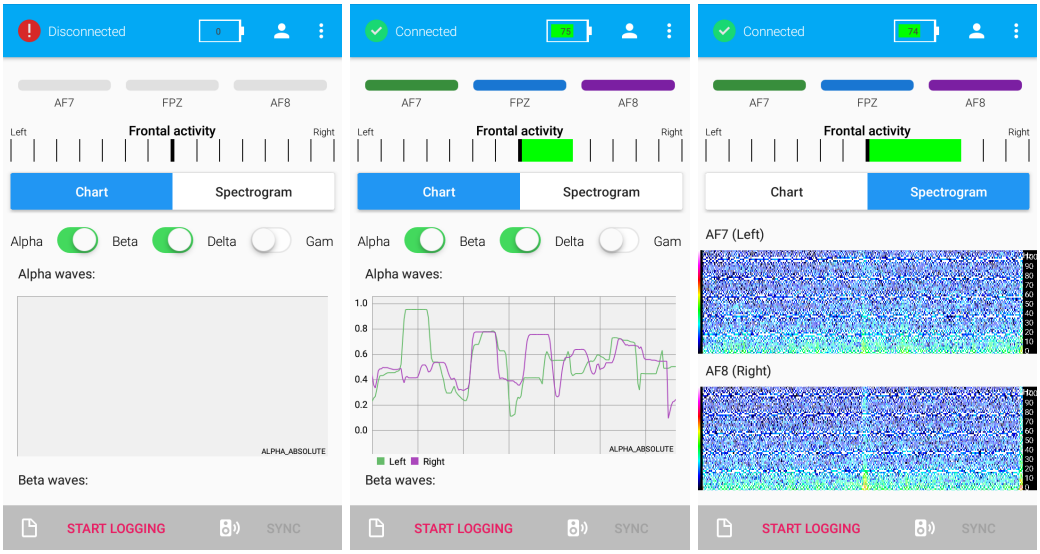
Pantalla principal:

La pantalla principal muestra la información más relevante y permite la conexión con la headband. La información que podemos encontrar en la pantalla principal es el estado de conexión de la headband, el estado de conexión de cada electrodo, la actividad frontal y una visualización de los datos que se están recibiendo en tiempo real.

A parte de esta información, encontramos en la *Toolbar* diferentes botones:

- El primer botón es el icono de usuario, y permite acceder a la pantalla del perfil del usuario.
- El otro botón son las opciones de la aplicación, que contiene dos submenús. Uno permite ir a la pantalla de Historial y la otra a los Ajustes.

Contiene además dos botones en la parte inferior de la pantalla, START LOGGING y SYNC. Estos botones son los encargados de gestionar el inicio y la parada de la grabación de datos, y de establecer la marca para la sincronización en los ficheros.

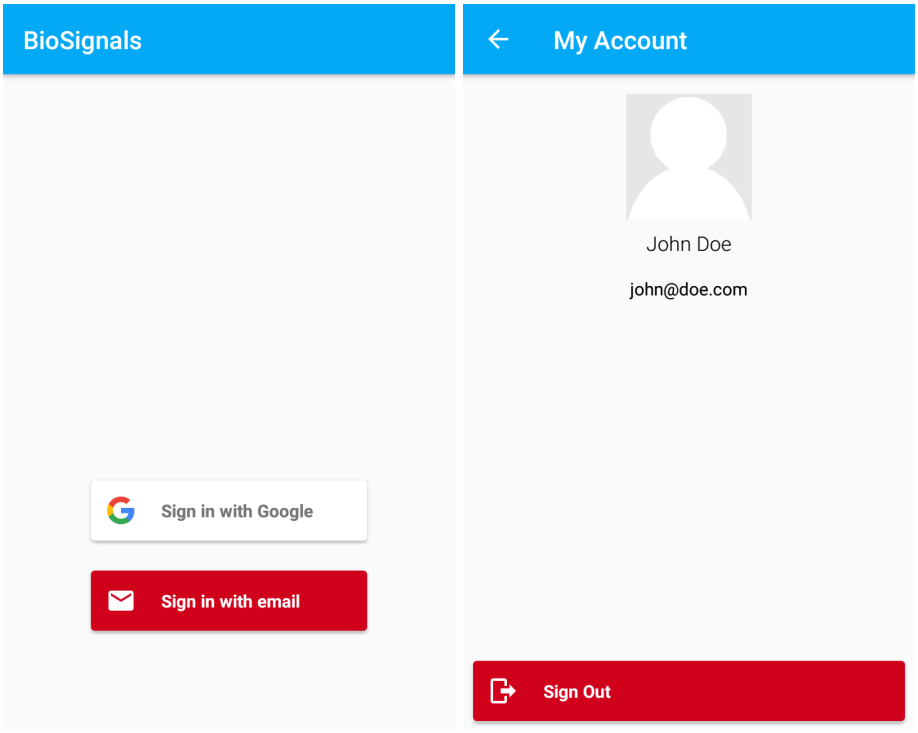


(a) Muse desconectada **(b)** Conectada mostrando gráficos **(c)** Espectrograma

Figura 40.: Los diferentes estados de la pantalla principal

Pantalla perfil de usuario:

Esta pantalla permite al usuario iniciar sesión utilizando el método que prefiera o, en caso de ya haber iniciado sesión previamente, visualizar su perfil y cerrar la sesión.

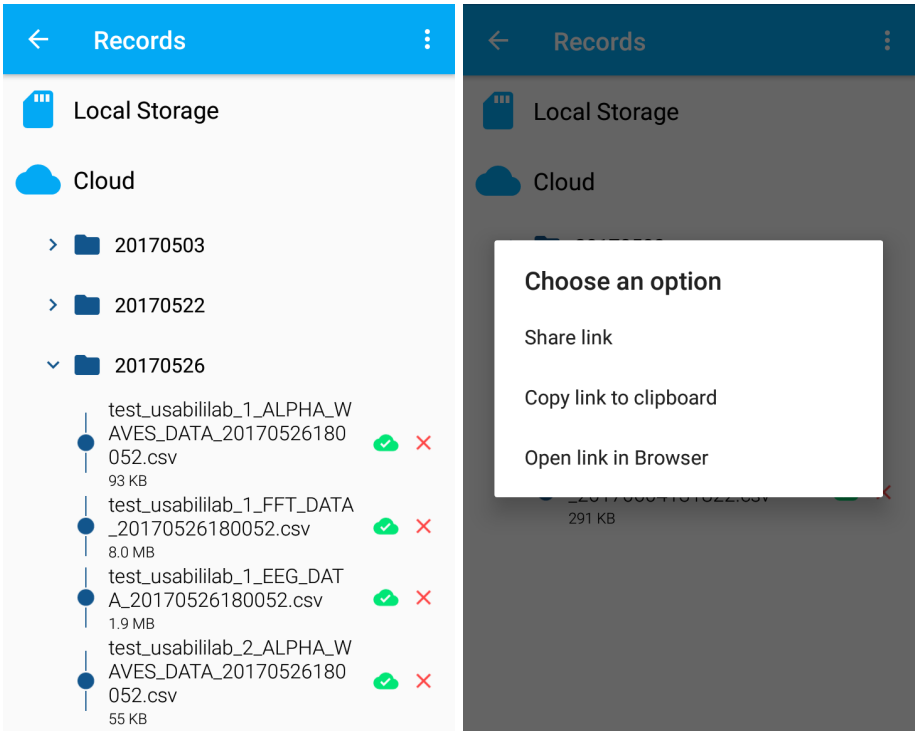


(a) Sesión no iniciada **(b)** Perfil de usuario John Doe

Figura 41.: Perfil de usuario. Diferentes vistas si se entra con un usuario logeado o no.

Pantalla historial:

En la pantalla historial podemos ver los ficheros de las distintas grabaciones realizadas. Permite el borrado de los archivos y la subida al cloud de los mismos. Además, ofrece la posibilidad de compartir un enlace a los ficheros que están en el cloud, para acceder a los mismos desde cualquier navegador.



(a) Historial de grabaciones **(b)** Menú para compartir

Figura 42.: La vista historial permite ver las diferentes grabaciones realizadas, así como gestionar las subidas al cloud.

4.2.7.2 *PowerBalanceView*

PowerBalanceView es una vista que ofrece, a partir de dos valores, una representación intuitiva de cual de los dos es mayor, y en qué proporción respecto a valores anteriores.

Para la implementación de esta vista, me he basado en el ejemplo que ofrece la documentación oficial de Android, Android Developers. [11]

Su funcionamiento es realmente sencillo. Necesita únicamente un valor para mostrar datos. Si el valor es positivo, la barra se llenará hacia la izquierda, indicando que el hemisferio más activo es el de este mismo lado. Si el valor es negativo, al contrario.

Se le han proporcionado capacidades de personalización, para poder adaptar los colores y tamaños fácilmente. Para ello se ha añadido al fichero res/values/attrs.xml el siguiente *styleable*:

```
<declare-styleable name="PowerBalanceView">
    <attr name="axisColor" format="color" />
    <attr name="axisSize" format="integer" />
    <attr name="backgroundColor" format="color" />
    <attr name="barColor" format="color" />
    <attr name="gradient" format="boolean" />
    <attr name="steps" format="integer" />
    <attr name="stepBarColor" format="color" />
    <attr name="stepBarSize" format="integer" />
</declare-styleable>
```

Figura 43.: Declaración del styleable

De esta forma se puede acceder a los mismos desde la declaración XML. Su utilización es muy sencilla:

```
<xyz.jbonet.powerbalanceview.PowerBalanceView
    android:id="@+id/balanceView"
    android:layout_width="match_parent"
    android:layout_height="24dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    app:backgroundColor="@android:color/transparent"
    app:steps="15"/>
```

Figura 44.: Declaración de la vista en la layout mediante XML

Para permitir parámetros por XML, debemos añadir además el constructor que admite el parámetro AttributeSet, que contiene los valores establecidos anteriormente en el Styleable.

Como esta vista ofrece una funcionalidad por si misma y su uso puede ir más allá de para lo que ha sido pensada inicialmente, he decidido separarla del proyecto y convertirla en una librería, además de liberarla bajo licencia *Apache 2.0*, accesible desde `https://github.com/jbonet/PowerBalanceView`. Así puede ser reutilizada en otros proyectos si es necesario e incluso ser modificada por terceros.

El resultado final de esta librería, con varias de las personalizaciones posibles lo podemos ver en la Figura 45 en la página siguiente.

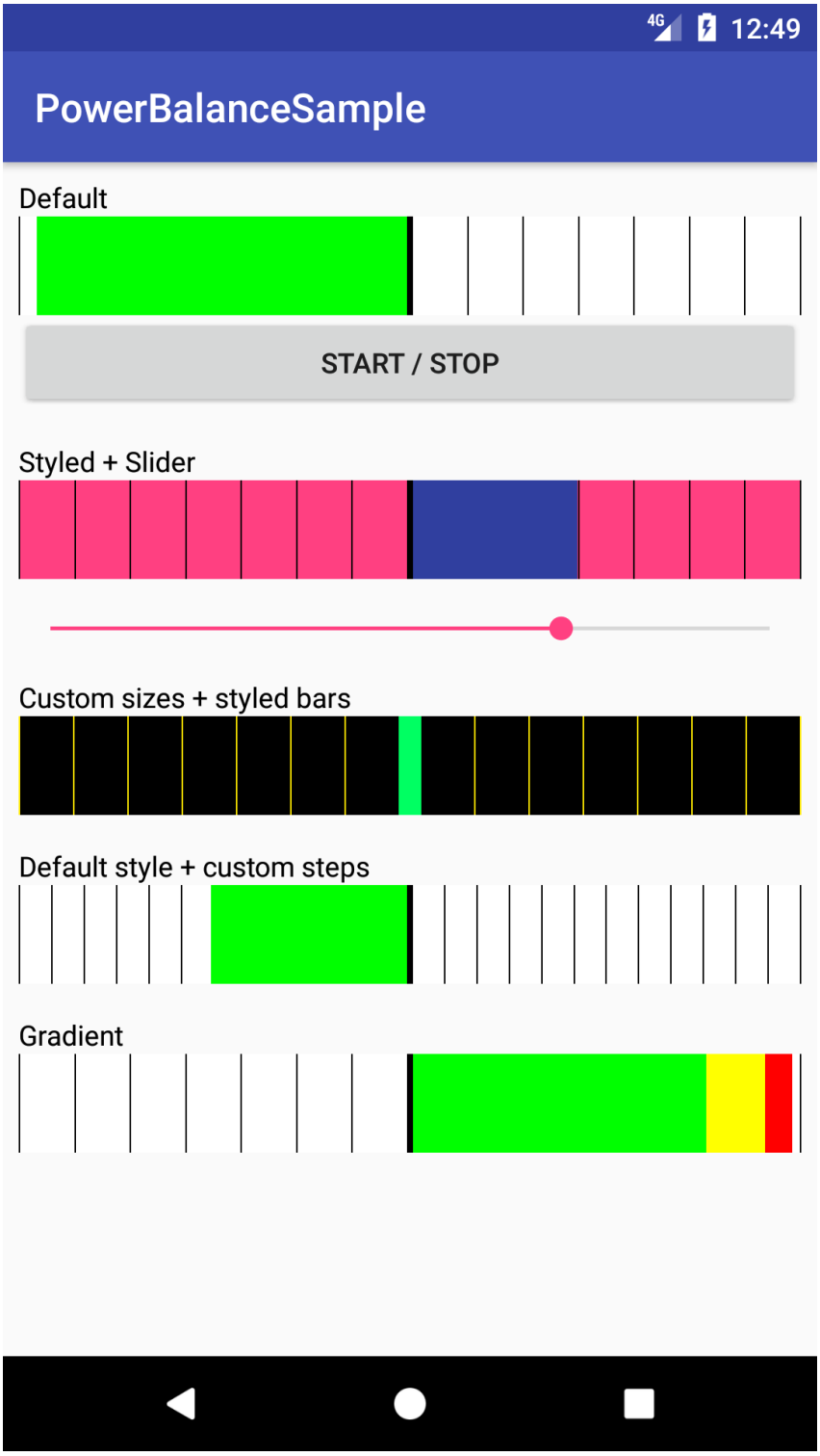


Figura 45.: Resultado final, con distintos estilos aplicados

5 BLOQUE DE FINALIZACIÓN

Una vez finalizado el desarrollo, procedemos a la realización de pruebas con usuarios, para comprobar si las mediciones que obtenemos son fiables.

5.1 PRUEBAS CON USUARIOS

Las pruebas con usuarios se han diseñado en base a las IAPS (International Affective Picture System), una base de datos de imágenes desarrollada para proveer un conjunto de estímulos emocionales. Estas imágenes tienen asociada una clasificación bidimensional (valencia y arousal), que permite cuantificar con precisión el efecto que causan sobre quien las mira.

5.1.1 Objetivo

El objetivo principal es observar la asimetría de las ondas alfa en la zona frontal al recibir un determinado estímulo. Esta asimetría determinará el impacto que ha causado el estímulo sobre el sujeto.

Otro objetivo es determinar si el dispositivo escogido para la medición es suficientemente preciso, o si en cambio, sufre de interferencias y no ofrece datos fiables.

5.1.2 Experimento

5.1.2.1 *Participantes*

Para la realización del experimento se reclutaron 9 personas, con edades comprendidas entre los 20 y los 27 años. En ningún caso los participantes habían sido expuestos previamente al conjunto de imágenes utilizado en la prueba.

5.1.2.2 *Materiales*

Para la realización del experimento se utilizarán diferentes herramientas:

- **Ordenador**, desde el cual se mostrarán las diferentes imágenes que conforman la prueba.
- **MUSE Headband**, dispositivo mediante el cual se realizará la medición EEG.

- **Smartphone Android**, necesario para ejecutar la aplicación desarrollada y recoger los datos obtenidos mediante la *MUSE Headband*.
- Conjunto de imágenes previamente cuantificadas (IAPS), para generar los estímulos en el sujeto.
- **Eye tracker**, para poder conocer en qué zonas de las imágenes mostradas se concentra la actividad de los sujetos.
- Pulsera **Empatica**, mediante la que obtener datos complementarios a los obtenidos mediante la MUSE, como el ritmo cardíaco.

5.1.2.3 *Procedimiento*

En el momento en que los participantes llegan al laboratorio son introducidos al experimento y se les presenta el documento del Consentimiento informado. Deben firmar este documento (adjunto en el Anexo A.1 en la página~50 para la realización del experimento. Durante la introducción al experimento se les debe informar de la naturaleza del mismo, pero sin detallar el contenido de lo que van a ver, de forma que no influya en los resultados. Se les explica también qué datos se van a grabar y cómo, y se les enseña cómo se coloca la headband.

Una vez explicado el procedimiento y el sujeto ha firmado el consentimiento, comienza el experimento.

El experimento consta de **dos** fases. En la primera fase el sujeto es expuesto a distintas imágenes que se clasifican en tres grupos: positivas, negativas y neutras. Cada imagen aparece durante 5 segundos en la pantalla.
Una vez finalizada la primera fase inmediatamente comienza la segunda fase.

En esta fase del experimento, a diferencia de la primera, lo que el sujeto observa es una sucesión de vídeos que contienen expresiones faciales. Los estímulos generados en esta fase podrán ser positivos o negativos, pero no se considera ninguno neutro como sí ocurría en la primera fase.

En total, el experimento consta de 55 elementos, entre imágenes y videos, que son mostrados durante 5 segundos cada uno. Así, el experimento tiene una duración de 275 segundos por sujeto. Un tiempo inferior a 5 minutos, de forma que el sujeto no lo considere demasiado largo y pierda la atención.

5.1.2.4 *Grabación de los datos*

La grabación de los datos durante el experimento se lleva a cabo mediante la aplicación desarrollada en este proyecto. Los datos pueden visualizarse en tiempo real mientras se realiza el experimento.

Los datos EEG grabados provienen de los sensores AF7 y AF8 según el sistema internacional 10-20 que podemos observar en la Figura 3.

La medición de la asimetría de las ondas alfa proviene de los datos proporcionados por el SDK de la propia headband, que se guarda en el fichero ALPHA_WAVES.

Se guardan varios ficheros más, proporcionando más datos sobre la realización del experimento. Estos ficheros contienen:

- La medición **EEG en raw**, para poder realizar un análisis más extenso sobre los datos.
- La **FFT** realizada en tiempo real sobre los datos.
- El valor de la **intensidad de las ondas ALFA** en cada uno de los sensores en cada instante de tiempo.

5.1.3 Resultados

Una vez obtenidos los datos con la realización del experimento, pasamos al análisis de los mismos.

Para el cálculo de la asimetría frontal alfa, hemos utilizado la diferencia entre los valores del canal izquierdo y los valores del derecho, de forma que si el resultado es positivo, la actividad alfa es mayor en el hemisferio izquierdo y viceversa.

Como cada imagen es mostrada durante 5 segundos, obtenemos el índice de asimetría durante una imagen en concreto, calculando la media aritmética de las diferencias de valores obtenidos durante estos 5 segundos.

$$\text{assymetry index} = \overline{\{\alpha\text{left}_1 - \alpha\text{right}_1, \alpha\text{left}_2 - \alpha\text{right}_2, ..., \alpha\text{left}_n - \alpha\text{right}_n\}}$$

Tras realizar el cálculo, obtenemos un índice de asimetría único para cada uno de los elementos. De esta forma podemos observar si el estímulo ha sido positivo o negativo de forma global para esa imagen.

Los resultados obtenidos para las 5 primeras imágenes son los siguientes:

Imagen	Sujeto1	Sujeto2	Sujeto3	Sujeto4	Sujeto5	Sujeto6	Sujeto7
1	0,320	0,261	0,076	0,167	-0,199	-0,123	0,354
2	0,369	0,198	0,193	0,071	0,014	0,272	0,200
3	0,308	0,151	0,153	0,081	-0,054	-0,027	-0,063
4	0,326	-0,004	-0,059	0,078	-0,048	0,091	0,006
5	0,402	-0,159	0,178	0,011	0,020	0,118	0,022

Tabla 3.: Resultados del índice de asimetría para las 5 primeras imágenes

Podemos observar que la mayoría de sujetos reaccionan de forma similar ante los distintos tipos de estímulos. Cuando la imagen mostrada es positiva, el índice de asimetría suele ser positivo, y en caso de que la imagen mostrada sea negativa, el índice de asimetría tiende a ser negativo.

Dado que el cálculo es la diferencia entre el valor de la actividad alfa en el hemisferio izquierdo y la del derecho, cuando el índice es positivo indica mayor actividad en el hemisferio izquierdo. Cuando el índice es negativo, indica una mayor actividad en el derecho. Como hemos visto anteriormente en la sección 3.2 en la página~11, la relación entre la actividad de las ondas alfa y la actividad cerebral es inversa, por lo tanto cuando:

- Índice $>$ 0: Actividad en el hemisferio derecho es mayor que en el izquierdo.
- Índice $<$ 0: Actividad en el hemisferio izquierdo es mayor que en el derecho.

Los resultados obtenidos chocan frontalmente con esta definición, ya que estamos observando que en los estímulos considerados como positivos se estaría activando el hemisferio derecho, encargado de procesar los negativos, y en estímulos negativos ocurre lo contrario.

La obtención de este resultado ha sido inesperada, lo que ha provocado una revisión completa de los resultados. En esta revisión se ha detectado una incoherencia en los datos que se repite durante todo el experimento.

La inconsistencia encontrada en los datos es que la diadema envía datos repetidos durante varios segundos, lo que causa una alteración de los resultados, no reflejando con fidelidad las ondas cerebrales del sujeto en ciertos momentos.

Para comprobar si ha sido un error puntual de configuración o interferencias en los electrodos, se ha repetido el experimento con tres sujetos utilizando una localización diferente. Este nuevo experimento ha comenzado con la carga de la batería de la diadema al 100 %. Una vez realizada la nueva prueba y analizados los datos obtenidos, se ha comprobado que el comportamiento encontrado en el experimento original ha vuelto a producirse. La diadema **no transmite los datos correctamente**.

5.2 CONCLUSIONES

Como conclusión del proyecto, se puede decir que se han alcanzado los objetivos marcados inicialmente, aunque me hubiera gustado más que los resultados obtenidos en las pruebas hubieran sido útiles. A pesar de esto, considero que ha sido una experiencia positiva, ya que el desarrollo de la aplicación se ha podido completar.

El hecho de desarrollar una aplicación desde cero me ha obligado a aprender a gestionar el tiempo, debiendo dividir el proyecto en tareas. Además he tenido que aprender nuevas tecnologías que no conocía, como RxJava. También he reforzado los conocimientos en áreas que ya conocía, como es el desarrollo Android. Es decir, he podido aprender cómo gestionar un proyecto de principio a fin, pasando por todas sus fases.

Un factor importante durante el desarrollo del proyecto ha sido la temática del mismo, ya que la Electroencefalografía era un área totalmente desconocida para mí. Conocer y aprender el funcionamiento de esta ha requerido varias lecturas y tiempo, pero finalmente ha resultado ser un campo muy interesante.

Respecto a las pruebas con usuarios, ha sido una lástima detectar el error en los datos que ofrece la diadema, ya que finalmente no hemos podido analizar los datos como hubiéramos querido.

Para concluir, estoy contento con el resultado obtenido: una aplicación robusta y práctica, que ha funcionado en un escenario real, aunque los resultados obtenidos en el experimento no hayan sido los esperados. Además, considero que haber adquirido conocimientos en áreas que de otra forma es posible que nunca hubiera llegado a interesarme por ellas, ha sido algo muy positivo.

5.3 POSIBLES TRABAJOS FUTUROS

Una vez finalizado este proyecto, podría continuarse de diferentes formas. En caso de continuar con el análisis de los datos, pueden plantearse nuevos proyectos a partir de los mismos, como pueden ser:

- Recreación del experimento, comprobando si hay alguna señal externa que hubiera podido alterar los resultados obtenidos, y utilizando un dispositivo MUSE diferente, pues podría ser un defecto de la diadema utilizada.
- Otro posible caso sería obtener datos sobre el sujeto, y poder detectar el estado en el que se encuentra: detectar concentración, emociones y otros estados de ánimo.

- También podría utilizarse en el campo de los videojuegos, de varias formas: Utilizándolo como controlador para realizar acciones, o para detectar el estado del jugador y personalizar la experiencia.

En otro caso, podemos continuar con el desarrollo de la Custom View creada para cubrir una necesidad. Ya que es una librería open source, puede ser utilizada en cualquier otro proyecto.

REFERENCIAS

[1] Robert E. Wheeler, Richard J. Davidson, and Andrew J. Tomarken. Frontal brain asymmetry and emotional reactivity: A biological substrate of affective style. *Psychophysiology*, 30(1):82–89, 1993.

[2] Claude E. Shannon. Communication in the Pressence of Noise. *Proceedings of the IRE*, 37(1):10–21, Ene 1949.

[3] Muse Developers. API Reference: MuseDataPacket. http://developer.choosemuse.com/android/android-api-reference#/classcom_1_1choosemuse_1_1libmuse_1_1_muse_data_packet.html, 2014. [Online; acceso 15 abril 2017].

[4] Piotr Wendykier. JTransforms. https://sites.google.com/site/piotrwendykier/software/jtransforms, 2015. [Online; acceso 18 abril 2017].

[5] Muse Developers. Available Data: Raw FFTs for Each Channel. http://developer.choosemuse.com/research-tools/available-data#Raw_FFTs_for_Each_Channel, 2014. [Online; acceso 15 abril 2017].

[6] Firebase. FirebaseUI for Android. https://github.com/firebase/FirebaseUI-Android, 2016. [Online; acceso 25 abril 2017].

[7] Google. The Tasks API. https://developers.google.com/android/guides/tasks, 2017. [Online; acceso 18 abril 2017].

[8] Philipp Jahoda. MPAndroidChart. https://github.com/PhilJay/MPAndroidChart, 2016. [Online; acceso 15 abril 2017].

[9] bmelnychuk. AndroidTreeView. https://github.com/bmelnychuk/AndroidTreeView, 2016. [Online; acceso 2 mayo 2017].

[10] Guillaume Adam. FrequencyView. https://bitbucket.org/galmiza/spectrogram-android/src/2d94f141359e94f930d843d307a9f37807a11466/app/src/main/java/net/galmiza/android/spectrogram/FrequencyView.java, 2013. [Online; acceso 16 abril 2017].

[11] Android Developers. Creating a View Class. https://developer.android.com/training/custom-views/create-view.html. [Online; acceso 22 mayo 2017].

[12] imotions. Frontal asymmetry 101. https://imotions.com/blog/frontal-asymmetry-101-get-insights-motivation-emotions-eeg/, 2017. [Online; acceso 3 junio 2017].

[13] Jing Chai, Yan Ge, Yanfang Liu, Wen Li, Lei Zhou, Lin Yao, and Xianghong Sun. *Application of Frontal EEG Asymmetry to User Experience Research*, pages 234–243. Springer International Publishing, Cham, 2014.

[14] Bahar Güntekin and Erol Başar. A review of brain oscillations in perception of faces and emotional pictures. *Neuropsychologia*, 58:33 – 51, 2014.

[15] Ross Avila. The relationships between frontal alpha asymmetry, mood, and emotional memory. *Graduate Theses and Dissertations*, 2011.

A ANEXO

A.1 CONSENTIMIENTO INFORMADO

Consentimiento informado para participar en la investigación de recogida de datos biométricos:

En cumplimiento de la Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal (LOPD), mediante el presente documento acepto que:

Gracias por su interés en participar en este experimento de percepción, donde se recoge actividad cerebral mediante una electroencefalografía (EEG),un dispositivo de lectura de constantes vitales y un eye-tracker que permite detectar el movimiento ocular.

Si tiene alguna duda sobre este experimento consulte al asistente antes de continuar.

Antes de continuar con el experimento, debe rellenar el cuestionario que encontrará más abajo y firmar su autorización para que podamos usar en usted las herramientas que hemos comentado antes: EEG, pulsera de medición de constantes vitales y eye-tracker.

Al firmar este formulario da su consentimiento a participar en el experimento y a que sea analizada su actividad cerebral mediante una EEG, se recojan sus constantes vitales y la respuesta galvánica de su piel y se registre su actividad ocular.

El investigador asistente que le acompaña le debe explicar todos los detalles que necesite del experimento. Su firma también confirma que ha recibido esta información.

Si usted se siente incómodo o no se encuentra bien, se puede interrumpir el experimento en cualquier momento.

Con su firma, reconoce saber que el uso de los electrodos puede provocar algunas marcas en la cara que se desvanecen de manera natural en algunos minutos; su pelo puede quedar húmedo después del procedimiento a causa del uso de la solución salina de la EEG.

El experimento solo dura unos minutos, en cualquier caso menos de media hora.

Lea detenidamente las siguientes preguntas médicas.

Si la respuesta a cualquier pregunta es "sí", debe indicarlo al investigador asistente para evaluar si puede participar en este estudio.

• ¿Está tomando o ha tomado recientemente, cualquier medicamento de venta libres?

• ¿Existe algún motivo que impida su visión normal, con o sin corrección?

• *¿Alguna vez ha sufrido de epilepsia?*

• *¿Ha tenido alguna cirugía en la que se haya colocado elementos metálicos han en su cabeza?*

• *¿Tiene usted un marcapasos instalado?*

• *¿Utiliza algún otro dispositivo médico?*

• *¿Se ha sentido mal durante los últimos días?*

• *¿Sufre de cualquier enfermedad crónica de la piel (por ejemplo, eczema, soriasis) ?*

• *¿Ha consumido alcohol o drogas en las últimas 24 horas?*

• *¿Actualmente tiene algún corte o abrasiones en la cabeza?*

- *¿Padece o ha padecido algún tipo de discapacidad auditiva?*

<u>Declaración del Participante</u>

Doy mi consentimiento informado para participar en este experimento de investigación que supone una sesión de grabación de la actividad cerebral con un Electroencefalograma, el registro de mis constantes vitales y la respuesta galvánica de la piel, la captación de mi actividad ocular y la grabación del experimento en vídeo y fotografía.

Soy consciente de que mi participación es voluntaria y que me puedo retirar en cualquier momento sin dar ninguna razón. Soy consciente de que toda la información dada por mí o los datos grabados de mí será manejada de manera confidencial.

Firma del participante _____

Fecha _____

<u>Declaración del investigador</u>

Creo que el participante ha sido plenamente informado sobre el experimento, sobre el procedimiento de registro de EEG al nivel necesario para que dé su consentimiento informado. He analizado todos los aspectos relevantes del procedimiento con el participante, y respondió a todas las preguntas de forma satisfactoria. He observado que el participante ha rellenado todas las secciones correspondientes de este consentimiento para participar en la grabación de EEG.

Firma Investigador _____

Fecha _____

Nombre de la institución _____